Towards Efficient Software Component Evaluation: An Examination of Component Selection and Certification

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

Abstract

When software systems incorporate existing software components, there is a need to evaluate these components. Component evaluation is of two kinds according to literature: component certification is performed by an independent actor to provide a trustworthy assessment of the component's properties in general, and component selection is performed by a system development organization. While this principle is in general understood, in practice the certification process is neither established nor well defined. This paper outlines the relationship between the evaluations performed during certification and selection. We start from the current state of practice and research and a) propose a component-based life cycle for COTS-based development and software product line development, b) identify a number of differences in process characteristics between the two types of evaluation, and c) classify concrete quality properties based on their suitability to be evaluated during certification (when there is no system context) and/or during system development.

1. Introduction

Software systems include more and more pre-existing components, either in the form of Commercial-Off-The-Shelf (COTS) software or product lines [5]. One of the most important conditions for a successful reuse is the fulfillment of functional and quality requirements and consequently the known properties of the selected components. This paper examines the principles of *evaluation of pre-existing components*.

Current literature distinguishes two main approaches to component evaluation: the evaluation carried out by a system builder when selecting components [14] and an envisioned component certification [1]. Figure 1 shows the relevant organizations and (main) processes involved in the component business: *component development organizations* develop components; *system development organizations* evaluate components with respect to system requirements and (if selected) use them when building systems; independent *component certification organizations* evaluates components according to standardized procedures, so that an issued certificate is seen as a quality stamp which increases the trust in the component.

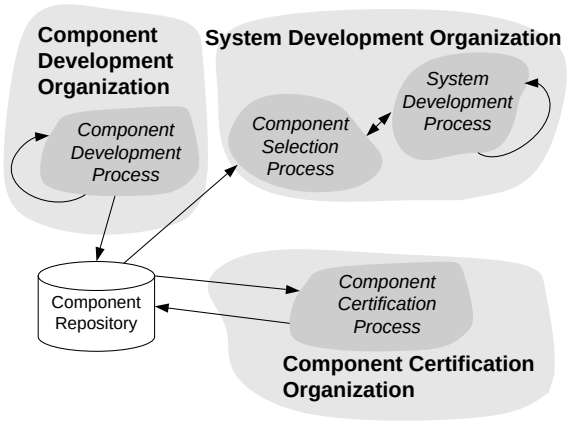

Figure 1. The main organizations and life cycle processes related to component evaluation.

However, by aligning these processes side by side, new questions arise which have not previously been answered. What similarities and differences are there between the evaluations performed during certification and selection? To what extent can the same processes and methods be used? (This is particularly relevant when the processes occur in the same organization, such as during product line development.) Which quality properties can and should be evaluated and assessed in isolation as part of certification? How can the processes complement each other, so that the certification results are useful for system developers when comparing and evaluating components with their particular context in mind? How can these processes interact to form an efficient overall process? This paper addresses these questions, and the contributions are: first, in Section 2 we identify principal differences in process characteristics of the evaluation performed during component certification and component selection. Second, in Section 3 we propose a classification scheme for quality attributes, based on whether they need to be evaluated with a particular system in mind, or if they can (to some extent) be evaluated during certification, i.e. when there is no system context. Third, in Section 4 we describe in more detail the overall life cycle with the four

processes of Figure 1, for two different business contexts: Commercial-Off-The-Shelf (COTS) software, and product lines [5]. In addition to these sections referred to above, the rest of Section 1 describes the research method and scope limitation, Section 5 describes the work related to this study, and Section 6 concludes the paper.

1.1 Research Method

This paper is the result of several research phases. We have previously separately studied existing literature and practice within the areas of component certification [1] and component selection [14], as well as the overall component-based life-cycle [8]. In this paper we have combined these results systematically, as the structure of the paper indicates. We have compared process characteristics such as goals, activities, inputs/outputs, roles, etc., and also examined existing definitions of quality characteristics in order to outline which of these can be evaluated during certification and which need a specific system context to be meaningfully evaluated. This paper should be seen as a theoretical examination and a proposal which will be used as a foundation for further empirical studies.

1.2 Scope Limitation

The component-based development processes (component development, component certification, system development and component selection) differ significantly in different organizational and business settings. Three different types of component-based development have been described [8]:

- **COTS-based development.** Organizations building systems use components available in the commercial marketplace, and have no direct influence on the component providers [30].
- **Product-line development.** A single organization develops components for internal reuse in several products [5].
- **Architecture-driven component development**. Components are developed as the result of a top-down system design process and not for reuse, but the component-based paradigm is adopted e.g. to enforce modularization and to be able to use the benefits of standard services of a component technology.

This paper is only concerned with evaluation of pre-existing components intended for reuse in systems partly unknown during the development of the component. Excluded is thus architecture-driven development, where evaluation means verification of correctness with respect to the component specification.

2. Examination of Process Characteristics

This section first outlines the process characteristics of the evaluation performed during component selection and certification based on literature reviews and an elaboration given the different goals of COTS-based development and product line development. Then these processes are compared, in order to address the questions asked in the introduction: What similarities and differences are there between the evaluations performed during certification and selection? To what extent can the same processes and methods be used? The whole section is based on a comparison of our previous separate surveys of the two fields [1, 14].

2.1 Characteristics of Certification

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 differences between COTS-based and product-line development are:

- **COTS-based development.** The component vendor orders a component certification, if it is considered beneficial from a business perspective. 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 thanks to the quality stamp the certification results represent (it could also be noted that there may be standards and regulations mandating certain evaluations).
- **Product-line development.** All processes occur within the same organization, but by different separate sub-organizations. A successful certification of a component would mean that some properties are known in general and thus can be reused with higher confidence in shorter time; an ideal division between certification and selection means evaluating enough during certification to make the overall process most efficient, but not more.

Figure 2a) shows how the component certification process takes a *component* and a set of *properties to evaluate* as inputs, and produces *information about the component*.

2.2 Characteristics of Component Selection

The evaluation performed during component selection is made with respect to some goals and objectives of the envisioned system as well as of process issues such as acceptable cost and risk. Variations in these goals affect for example how many components should be evaluated, how thoroughly they need to be tested, which properties should be evaluated, etc.

- **COTS-based development.** 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 [20], from many potential components to fewer which – with some confidence – are believed to suit the system requirements best.

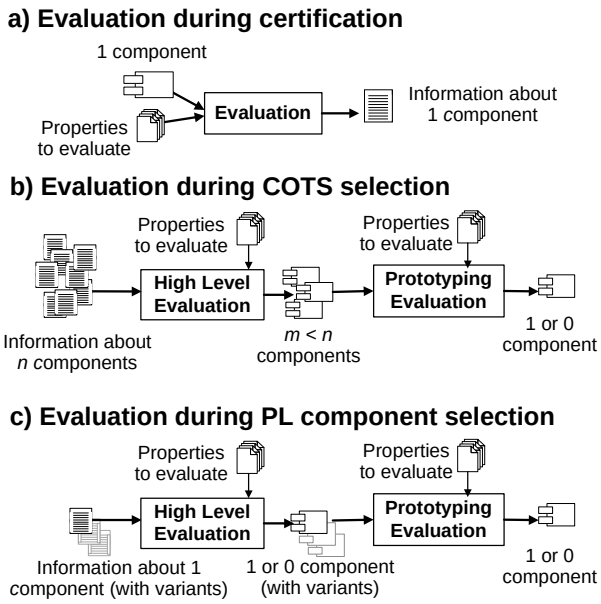

Figure 2. The certification and selection activities. Notation according to SADT [22].

- **Product-line development.** When developing or evolving a product in a product line, typically only one pre-existing component is evaluated, namely the one developed internally. The goal for this evaluation is to assess its suitability for a specific product, and if the component is not sufficient the outcome is that it needs to be changed before the system can be built. (In case there is no pre-existing component a new component development process is initiated, or possibly a COTS component or subcontracting could be considered.)

For both COTS-based development and product line development, the evaluation can be divided into two phases: During *high-level evaluation* only information about components is used, and during *prototyping evaluation* the actual component is a tested and prototyped [14]. Any certification results would be an important input to the high-level evaluation, since this information is considered more trustworthy than for example the vendor's marketing material. Figure 2b) describes the progressive filtering of COTS components between these two phases, and Figure 2c) describes how a single component, with some variants, is evaluated during product-line development. By specifying *properties to evaluate* the concrete evaluation procedures are determined.

2.3 Fundamental Process Differences

The fundamental principal differences between the evaluations performed during component selection and certification are:

Source of properties to evaluate. In the case of component selection, properties are derived from (or at least, related to) system requirements, while properties to be certified are prescribed by the component vendor and/or standards and regulations.

Availability of component. 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).

Goal of evaluation. 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. As a consequence of the difference in goals, there are several other differences, as follows:

Level of confidence required. During component selection, evaluation only needs to last until the evaluator has enough confidence to make a selection (i.e. the required rigor of the evaluation is related to the criticality of the system being built), while for component certification the confidence needs to have some objective and comparable meaning.

Flexibility of evaluation. When the goal is to select the component that best fits a specific system the evaluation can be very flexible and opportunistic. This means that at each point in time one asks 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. (This is noticeable in the available selection methods PECA [6] and PORE [16, 19].) For component certification, the goal of the evaluation does not change during the evaluation, so no major changes in the process are expected during process execution. In fact, the standardized process is one part of the strength of an issued certificate.

Outcome. During component selection the most important result is the decision to use a certain component or not, while during component certification, the documentation of the evaluation is the most important outcome, perhaps in the very condensed form of a "certificate".

From this list it follows that from a process point of view the component evaluation is carried out very differently. Thus, when defining details of these processes we expect many things to differ: the planning of the evaluation, when to stop evaluating, document templates, roles, etc. However, it is apparent how they fit together in a sequence through the *information about component*, which is the output of the certification activity and, if existing, is an important part of the information used as input to high-level evaluation during component selection (see Figure 2). It is apparent that the format of this information is an important area for standardization in the COTS context. For product-line development

the selection process and certification process should be defined and improved together, since they are subordinated the same overall goals of the same organizations, and changes in one may well affect the other.

It should also be noted that nothing so far prevents that the actual methods employed to evaluate certain component properties could be identical, such as a specific benchmark performance test.

3. Examination of Quality Characteristics

This section addresses the question asked in the introduction "Which quality properties can and should be evaluated and assessed in isolation as part of certification?" We have studied the 6 quality characteristics and 27 sub-characteristics defined by the ISO/IEC standard 25000 [11]. In addition we have also studied the IEC standard 61508 [12] describing functional safety. This section first shows the different challenges of evaluating different characteristics, after which a crude classification of characteristics is presented, based on their suitability for evaluation during certification and/or selection.

3.1 Example Quality Characteristics

Due to space limitations, we have selected a few quality characteristics which we consider being of general interest for many systems and widely understood, and also useful as pedagogic examples:

Accuracy (a sub-characteristic of functionality) concerns the software's ability to provide of right or agreed results or effects. Accuracy includes number of significant digits and rounding treatment, which can be meaningfully evaluated without a system context, and can thus be part of both component certification as well as component selection. Accuracy also includes failure ratio, which measures the number of erroneous test results in relation to a set of tests; this includes a subjective selection of test cases which is less than ideal for component certification – the selected test cases may or may not correspond to actual usage.

Compliance (also a sub-characteristic of functionality) concerns the software's adherence to relevant standards; this can be evaluated during certification and should be an important part of a certificate.

Suitability (also a sub-characteristic of functionality) includes the software's coverage of the desired functionality, which can clearly not be evaluated without a system context, i.e. not be certified. This is a very important evaluation criterion during component selection however.

Efficiency (includes time behavior and resource behavior) 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) which makes it very difficult to provide the type of general guarantees which one expects from a certificate. However, it may be possible to package assertions or test results with information about the test environment attached, which requires a standardized format, or even better a standardized environment would be used. Another possible solution is to parameterize the results, as has been suggested for component worst-case execution time where a partial evaluation without system context is performed and packaged parameterized on input ranges [9].

Safety is a system property which depends on the external environment context [12] and it is not meaningful to discuss neither the safety of a component (it is the system as a whole which is safe or unsafe), nor the safety of software in isolation. Nevertheless, it is desirable to be able to provide some type of safety-related guarantees for general reusable software components. The approaches adopted by IEC 61508 [12] are to specify (types of) methods, tools and processes to be used during development, and we can also note that safety-related functions are considered (not components). In addition to these two approaches, it is possible to discuss verification and certification not of component safety directly, but of various properties related to safety. One important example is *reliability* with sub-characteristics such as *fault tolerance*, *recoverability*, and *availability*; the discussion for these properties is similar as for accuracy above: they can to some extent be verified objectively but there is also a system- and usage-dependent part. A component certificate could nevertheless in principle be used as a foundation for further system-specific safety analysis during the selection process and also for system validation further downstream. For example, a component certified to be very reliable could be a potential candidate.

3.2 Classification of Quality Characteristics

Based on the above examination of concrete quality characteristics and sub-characteristics, and indicators of these, we here propose a classification scheme for the indicators of characteristics:

- **Objective.** Some properties can be objectively measured and guaranteed in general, and are thus suited for certification. This applies to some indicators of for example accuracy, reliability, and compliance to standards.
- **Usage-dependent.** Some properties are closely tied to the usage of the component, such as suitability and some indicators of accuracy. Any evaluation without a system context needs to make assumptions on usage in order for the evaluation to be relevant, and the results are only valid within these assumptions. For example, assumptions behind test cases must be well motivated and specified. Ideally such hypothetical usage profiles

would be standardized, thus making it easy to compare components during high-level evaluation. With more research and standardization, it could be possible to package the results parameterized on usage as we have described for worst-case execution time [9]; as a minimum the assumptions must be provided with the results.

- **Environment-dependent.** Some properties vary with the technical environment, the most apparent example probably being efficiency. Certification would be similar as for usage-dependent properties in that the assumptions on which the evaluation is based need to be openly published with the results, and preferably are standardized.
- **System-wide.** Some properties cannot even be fully evaluated with a system context, unless the whole system is already available. Examples are safety, which is an inherently system-wide property, and to some extent efficiency, since e.g. the execution time of one component may depend on the execution time of other (seemingly unrelated) components due to cache behavior and similar.

4. Life Cycles

This section gives a high-level view of the life cycles of COTS and product-line products, thus presenting a more detailed version of Figure 1 in the light of the discussion in the previous sections.

4.1 The COTS-Based Life Cycle

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. There is 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 life cycle model is shown in Figure 3; the model is intended to be general enough to cover many types of development models, both sequential, iterative, and evolutionary (the arrows denote data flow). In the figures we therefore present activities as (sub)processes which could be carried out in parallel, in sequence, iteratively in an agile fashion, etc. Component development includes *requirements engineering* for the component, *design* and *implementation* using some architectural standards and considering some environment constrains. The component is *verified* internally and *released*. The component is then delivered to the certification organization. Based on information about a component such as documents, source code, tutorials, examples, and so on, and also depending on the domain, the certification organization performs a set of activities to verify the component properties and to issue the certificate. According to [1], these activities are: *data collection* (of information about the software component and the environment); *define, design and plan* (that is, specifying techniques and methods and planning the evaluation); and *evaluation* (collecting data and providing recommendations). According to the classification scheme presented above, objective quality characteristics or indicators are suitable for evaluation. It is also possible to evaluate usage- and environment-dependent properties with certain openly published usage profiles and environments (ideally, these are standardized).

The components are stored in the component vendor's *internal repository*. Some products and versions may be published (indicated in Figure 3 as a *public repository*). Certified versions of a component can be made public as well as newer, not yet certified versions. With "making public" we do not necessarily mean making the component itself immediately available, but in many cases making the information about the component public.

System development organizations may now find components. They are first subject to *high-level evaluation*, to which the certification results is an important input (as described in section 2.3). The description of objective properties can be easily compared with the system's requirements on the component, and the usage- and environment-dependent properties could be compared with the expected usage and environment of the component. Also evaluated are business considerations such as available support and vendor reputation. Some components will then be subject to *prototyping evaluation*, where prototypes are created to gather more detailed data, and with higher confidence, about the functionality and quality, and identify any architectural compatibility issues. Any knowledge gathered during both evaluation phases may be fed back to the *requirements engineering* and *design* activities, in several iterations if needed. Finally a component is selected and used when building the system.

When a component is selected and integrated into the system, the system development organization stores the components and evaluation information in an *internal repository* to allow future reuse in other projects. It then enters a mode of *maintenance and evolution* the system which may include integrating newer versions of the component in the future.

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 features of existing components. It could be noted that for software components, many of these requirements are closely related to design and system integration,

such as what platform and component technology the component is designed for.

4.2 The Product-Line Life Cycle

Product line components are intended to be reused in several systems, to address the needs of the product development. Since all processes take place within the same organization they can be better synchronized and performed more efficiently.

In the product-line life-cycle, shown in Figure 4, *system development* can impact *component development* much more directly, since system requirements can be forwarded directly to the component development. The *component selection* process is mainly used to evaluate suitability of the one and only one component developed internally for reuse in product line products. Typically external components should not be integrated into the system when there is a strategy to use an internal component for this purpose. As for the internal components, some may come in variants aimed at different types of products in the product line, and the selection process needs to select which of these to use. For example, there may be a resource efficient variant with limited functionality intended for low-end products and a richer variant intended for expensive products. If desired, the *evaluation* can be performed early during component development. Also, component *requirements* may be modified during component development based on changed or added system *requirements*; this scenario must be well-managed though, in order to avoid other problems of component development (e.g. late delivery) which may have consequences for other products in the product line.

The certification process would be carried out internally by a sub-organization, due to several reasons: first, with the same overall goals as component development and system development, certification would only evaluate properties of interest to the organization; second, for usage- and system-dependent characteristics test cases would be chosen based on the known usage and system context of the component (e.g. throughput and latency for a certain common input and envisioned number of simultaneous users, using the actual hardware used by products); third, in this way the organization's business/technical goals and knowledge are kept secret; fourth, certification is made efficient through close contact with the component development organization, and can use compatible methods and tools.

However, all this may possibly compromise objectivity. If this is perceived as a risk, a product line organization may let some components be certified by an external organization (there may also be regulations requiring this). We can conclude that product line organizations may be more efficient, accurate, and useful than certification of COTS, but less independent; a certification of a component may be seen as a standard verification of the component.

If a more precise evaluation of some properties is required during the component selection (*prototyping evaluation)*, this task should be given to the component certification sub-organization, which afterwards stores the new test cases (and their results). In this way, certification within a product line organization can be characterized as:

- **More customized** to the specific needs of the organization, compared to COTS certification by a third party. This effectively means that the distinction between system-independent and system-specific evaluation in practice disappears.

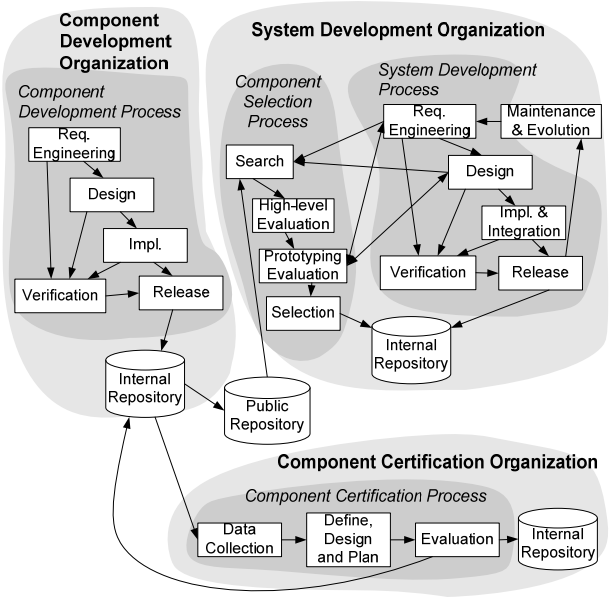

Figure 3. The COTS-based life cycle.

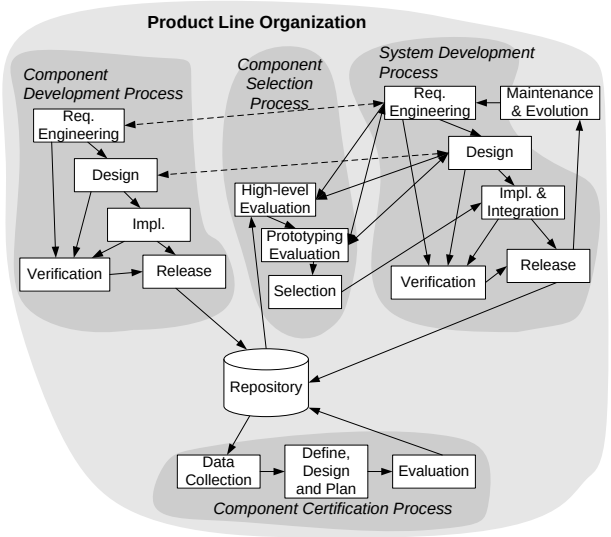

Figure 4. The product line life cycle.

- **More dynamic** than for COTS certification, since new needs may arise during system development and component selection which requires new evaluations to be performed, which then becomes part of the standard certification procedure in the future. This means that a component variant which previously was certified may at some later point in time no longer fulfill the certification criteria.

5. Related Work

To our knowledge, there are no publications combining the current knowledge of component selection and component certification as an integrated part of component-based development (CBD) process. Several authors discuss the lifecycle process divided into (i) *component development*, (ii) *component selection* and (iii) *system development* processes [3, 8, 15, 16, 19], but component certification is treated implicitly or not at all. There is some literature that relates component-based development processes to a certification process. In [27] the cooperation between component development and component certification are considered, and in [26] it is discussed whether a certification process is an (un)necessary part of component-based development. We relate 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.

SEI organized a workshop in 1997, where many of the principles of component selection were first documented [20], such as the typical progressive filtering of components which occurs during the evaluation, a principle which has been further elaborated in many of the published methods (see e.g. [6, 13]). Another principle is that of puzzle assembly [20], i.e. the evaluation of combinations of components together (see e.g. [4, 23]). Several methods suggest a close interaction between requirements engineering and component selection [3, 5, 15, 16, 19]. Taken together, the available literature points to four types of criteria to be evaluated: functionality, non-functional (quality) attributes, architectural compatibility, and business considerations such as vendor reputation and available support. We use the simple process model of [14] in this paper, which is a consolidation of existing selection methods to date. For a more lengthy discussion of existing literature on component selection we refer to our survey [14].

The idea of component certification is to ensure that software components conform to well-defined standards; based on this certification, trusted assemblies of components can be constructed [7]. There has been a line of research building on modeling of e.g. reliability [21, 32], and testing approaches such as a combination of black-box testing and fault injection [28], and a method for systematically increasing dependability scores by performing additional test activities [29]. A related approach is to supply tests in a standardized, portable format and let the integrator determine the quality and suitability of purchased software [18]. Meyer coined the term "low road" to signify research aiming at the definition of a component quality model to enable certification of existing components; the "high road" involves research aiming at production of components with fully proved correctness properties [17]. This second line of research is pursued by the SEI and its Predictable Assembly from Certifiable Components (PACC) initiative, where the goal is to enable certification by building theories and technologies which enable prediction of component assemblies based on known component properties [10, 24, 31], so that certification results can become the desired type of guarantee. Some literature outlines a component certification process [1, 7, 27] and in this paper we use the model presented in [1]. For a more extensive summary of component certification refer to our survey [1].

6. Conclusion

We have presented an overall view of the processes involved in the component-based life cycle, with a focus on the component evaluation performed during component selection and component certification. Our analysis reveals a number of fundamental differences between the two types of evaluations. Some of the differences are related to the process: the properties to evaluate come from different sources, the actual component is not initially used in the selection process, and the goals, flexibility allowed, level of confidence required, and type of outcome are all different. When examining concrete quality properties to be evaluated, we have identified four classes of properties: objective properties that can be guaranteed in general, usage-dependent and environment-dependent properties (which can be partly evaluated without a system context), and system-wide properties. Thus, certification can never cover everything which is important for a system developer, but if introduced it has the potential to improve the overall efficiency of component evaluation. In order to achieve this, a proper division of the evaluation must be found, and also issues of standardization, costs, and liability must be solved. Further research in component certification needs to consider how the evaluation results can be used by systems builders.

The research presented in this paper is limited to a theoretical examination, which contributes to an understanding of these processes, and will be used in our ongoing studies of component-based processes and reuse. This includes industrial collaborations

and empirical studies, at the C.E.S.A.R. center[1] in Brazil, where we are part of a large effort to establish component certification [1, 2], and the PROGRESS centre[2] in Sweden.

Acknowledgements

This work was partially supported by the Swedish Foundation for Strategic Research (SSF) via the strategic research centre PROGRESS.

References

[1] A. Alvaro, E.S. Almeida and S.R.L. Meira, "Software Component Certification: A Survey", *The 31st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Component-Based Software Engineering (CBSE) Track*, Porto, IEEE, 2005.

[2] A. Alvaro, E.S. Almeida and S.R.L. Meira, "Component Quality Assurance: Towards a Software Component Certification Process", *International Conference on Information Reuse and Integration (IRI)*, Las Vegas, USA, IEEE, 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*, LNCS 3412, Springer, 2005.

[5] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison Wesley, 2001.

[6] S. Comella-Dorda, J. Dean, E. Morris, and P. Oberndorf, "A Process for COTS Software Product Evaluation", *1st International Conference on COTS-Based Software System (ICCBSS)*, LNCS 2255, pp. 4-6, 2002.

[7] B. Councill, "Third-Party Certification and Its Required Elements", *4th Workshop on Component-Based Software Engineering (CBSE)*, Canada, IEEE, 2001.

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

[9] 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.

[10] S. A. Hissam, G. A. Moreno, J. Stafford and K.C. Wallnau, "Enabling Predictable Assembly", *Journal of Systems and Software*, 65(3), pp. 185-198, Elsevier, 2003.

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

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

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

[14] R. Land, L. Blankers, M. Chaudron, I. Crnkovic, "COTS Selection Best Practices in Literature and in Industry", In 10th International Conference on Software Reuse, Beijing, China, Springer, 2008.

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

[16] N. A. Maiden, C. Ncube, "Acquiring COTS Software Selection Requirements", *Software*, 15(2), pp. 46–56, 1998.

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

[18] J. Morris, G. Lee, K. Parker, G.A. Bundell and C.P. Lam, "Software Component Certification", *Computer*, 34(9), pp. 30-36, IEEE, 2001.

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

[20] P. Oberndorf, L. Brownsword, E. Morris, C. Sledge, *Workshop on COTS-Based Systems*, CMU/SEI-97-SR-019, Software Engineering Institute, 1997

[21] J. Poore, H. Mills and D. Mutchler, "Planning and Certifying Software System Reliability", *Computer*, 10(1), pp. 88-99, IEEE, 1993.

[22] D. T. Ross, "Structured Analysis (SA): A language for communicating Ideas", *IEEE Transaction on Software Engineering*, 3(1), pp. 6-15, January, 1977.

[23] G. Ruhe, "Intelligent Support for Selection of COTS Products", *Web-Services, and Database Systems: NODe, Web- and Database-Related Workshops*, Erfurt, Germany, LNCS 2593, Springer, 2003.

[24] H. Schmidt, "Trustworthy components: compositionality and prediction", *Journal of Systems and Software*, 65(3), pp. 215-225, Elsevier, 2003.

[25] Seacord, R., Bass, M., "Building Systems from Off-the-Shelf Components", *Software Architecture in Practice Second Edition*, Addison Wesley, 2003.

[26] 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.

[27] M. Tziakouris, A. S. Andreou, "A quality framework for developing and evaluating original software components", *Information and Software Technology*, 49(2), pp. 122-141, 2007.

[28] J. M. Voas, "Certifying Off-the-Shelf Software Components", *Computer*, 31(6), pp. 53-59, IEEE, 1998.

[29] J. M. Voas and J. Payne, "Dependability Certification of Software Components", *Journal of Systems and Software*, 52(2-3), pp. 165-172, Elsevier, 2000.

[30] K. Wallnau, S. Hissam, R. Seacord, *Building Systems from Commercial Components*, Addison Wesley, 2002.

[31] K. C. Wallnau, *Volume III: A Technology for Predictable Assembly from Certifiable Components*, Technical Report CMU/SEI-2003-TR-009, Software Engineering Institute, 2003.

[32] C. Wohlin, B. Regnell, "Reliability Certification of Software Components", *5th International Conference on Software Reuse (ICSR)*, pp. 56-65, IEEE, 1998.

[1] http://www.cesar.org.br/english/

[2] http://www.mrtc.mdh.se/progress/