Setting a Process to Effectively Measure COTS Functional Suitability

Alejandra Cechich
Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Neuquén, Argentina
Tel. (+54) 299 4490 312 – Fax: (+54) 299 4490 313 Email: acechich@uncoma.edu.ar

Mario Piattini
Escuela Superior de Informática, Universidad de Castilla-La Mancha, Ciudad Real, España
Email: Mario.Piattini@uclm.es

Abstract. In attempting to build a COTS integrated system, selection of candidates typically pays attention to specify search criteria and goals to be met. Yet they often overlook some elements in the process such as fact-based decisions and teamwork, which might drive the process helping increase the probability of success. In this paper, we identify some features that we have found useful in our research, and which we believe need further discussion before integrating a process for COTS components measurement.

1. Introduction

The particular nature of COTS components (black-box binary entities developed by third parties, and independently deployable in unforeseen contexts) imposes specific quality mechanisms to be put in place for their effective integration in the development life cycle of software applications. Firstly, components need to be properly documented and offer measurement mechanisms in order to be assessed and evaluated for selection. Secondly, both the quality and some of the extra-functional properties of the final (composed) system heavily depend on the individual properties of its constituent components, and therefore some traceability between the quality attributes at these two levels is required. Finally, the use of third-party COTS components may also introduce risks, such as potential harmful side effects to the system, or quality degradation of the final product. COTS component testing and certification may offer partial solutions to these problems [1].

The objective of our work is to enable composers to use some COTS components aspects as inputs to compute functional suitability and functional adaptation measures, which can be used to measure the impact of integration of COTS components on a stable system [2][3].

Of course, this objective is not unique to our work. It could, in fact, be considered a final goal composed of several sub-goals of software engineering. For example, in the realm of software requirements engineering, a significant body of research has accumulated under the general heading of COTS-Requirements Engineering. COTS-Requirements Engineering implies dealing with the use of third-party COTS components, whose services are balanced against system's requirements [4][5]. In that way, developing software becomes a matter of balancing required and offered functionality among the parties. But required functionality is highly dependent on component's users, i.e. stakeholders of a COTS selection process. Inputs to this process should include discussions with composers, reuse architects, business process coordinators, and so forth. Therefore, stakeholder's preferences should also be balanced before starting a selection procedure, so committed requirements are used to determine the degree in which a COTS candidate satisfies the required functionality [6][7].

However, an aspect that needs further discussion is the possibility of establishing a set of main stakeholders or roles for a selection process. Furthermore, when an organisation implements the approach, it needs to identify which specific roles and priorities it should address and what does or does not work for that organisation.

Another aspect that needs more attention is the diverse possibilities of documenting the required functionality [8]. In our proposal, we have chosen scenarios because of their wide use on evaluating architectures, however other representations might be more suitable depending on specific constraints of the system.

Our position is that the aspects and measures for functional selection can be combined into a measurement process, using a standard improvement framework – such as a Six Sigma process [9] – which guides and controls the whole activities [10]. However, elements to be used needs further discussion. Some features to be concerned with are defining and measuring a stable system (Sect. 2); balancing stakeholder's preferences on functional requirements (Sect. 3); and determining the architectural impact of a selection (Sect. 4). We conclude with some challenges implied by those activities, such as setting the right requirements for an application, collecting measures when dealing with incomplete and ambiguous information of COTS components, and son forth (Sect. 5).

2. Defining and measuring a stable system

As the selection process implies, the software is packaged with the goal that it will be used by many different systems. To introduce our approach, two kind of systems are referred to as follows:

- *The source system*. We refer to a source system as a set of pieces of software that running all together satisfy part of the user's requirements expressed as functional and non-functional properties, whose quality is able to be measured.
- *The Component-Based System (CBS).* We refer to a component-based system as a system that uses at least one component in conjunction with other components, legacy systems, and other pieces of software – including COTS components – to satisfy user's requirements. The concept of CBS is introduced to emphasize the fact that the output from the system satisfies the user's requirements by using the functionality supplied by at least one COTS component.

System satisfaction can refer both CBS and the source system – not necessarily both CBSs. In a Six Sigma-based process, the "define" phase sets the goals and context of the project. This phase is concerned with identifying key stakeholders, and determining their needs and criteria for selection. Some of the activities are as follows:

- Given a source system, we should specify the current degree of *satisfaction*/stability of the system under evaluation by defining goals, constraints, component context, domain as well as product's quality attributes.
- Then, we should identify sub-domains and relationships, i.e. relationships among pieces of software from an architectural point of view – legacy systems, COTS and standard components – and relationships among sub-domains.

The "measuring" phase is concerned with assisting the selection process by creating a complete description of the COTS candidates. Some of the activities include (1) measuring COTS components to determine their suitability with respect to the required services; and (2) assessing the impact of the integration of COTS components on a stable system.

Firstly, to measure COTS components, we have adapted the model in [11], which explores the evaluation of components using a specification-based testing strategy, and proposes a semantics distance measure that might be used as the basis for selecting a component from a set of candidates. This model supposes that there is an architectural definition of a system, whose behaviour has been depicted by scenarios or using an architecture description language (ADL). Our proposal [3], has adapted the model as a basement for quality measurement. We express the semantics distance in

terms of functional suitability measures, which provide a better identification of the different COTS component functionalities. Then, a system can be extended or instantiated through the use of some component type. Due several instantiations might occur, an assumption is made about what characteristics the actual components must possess from the architecture's perspective. Thus, the specification of the architecture *A* (*SA)* defines a specification S_C for the abstract component type *C* (i.e. $SA \Rightarrow S_C$). Given a specification S_C for an abstract component type *C*, a candidate component *K* to be a concrete instance of *C* must conform to the interface and behaviour specified by S_C. Although the process of selecting a component *K* consists of evaluating interface and semantic mappings, in our work only semantic mappings are addressed. Mappings in S_C, which represent the different required functionalities, are established between input and output domains. We focus on incompatibilities derived from functional differences between the specification in terms of mappings of a component K_i (S_{Ki}) and the specification in terms of mappings of S_C.

On the other hand, the degree in which a component solution needs adaptation might also be measured. During the mapping, it can be the case that the semantics of *K* are not sufficient for S_C. In this situation, *K* is somehow lacking with respect to the behavioural semantics of *C*. To overcome this, a semantic adapter must be specified (and built) such that, when composed with S_K, the adapter yields a component that is compatible with *C*. This situation allows us to define some measures to determine the functional adaptability of a component-based solution.

3. Balancing stakeholder's preferences

On the other hand, committing requirements for COTS component selection can be problematic. Traditionally, the requirements elicitation techniques have widely used a family of goal-oriented requirements analysis (GORA) methods. Our proposal extends a version of a Goal-Oriented Requirements Analysis Method called AGORA in [12] considering additional features of COTS components.

Initial goals, as considered in AGORA graphs, are the needs of the customers that will be refined and decomposed into sub-goals one after another. Therefore, with the functional goals of the component specified by mappings in S_C, the next step is to refine the goals considering the perspectives of different stakeholders – reuse architect, certifier, business process coordinator, etc. Then, the computation of stakeholder's preference values for the refined goals will allow us to add preferences to mappings of S_C. In our context of component-based systems, an AGORA graph describes the abstract specification of a required component (S_C) according to the scenario *S* [7].

When incorporating COTS components, goals should be balanced against COTS services. For example, using the main concepts of goal-oriented requirements engineering, the goal acquisition and specification process [4] includes the necessity of identify goals that help to distinguish between products (called core goals) from those that are provided by most available products (called peripheral goals). Besides the classification of goals as core and peripheral, the attributes desirability (level of importance for a goal to be met), and modifiability (level in which a goal can be modified) are proposed as attributes for goal description when selecting COTS components.

By using an AGORA graph, we can estimate the quality of several properties of the adopted goals. Particularly, correctness is assumed as a quality factor that represents how many goals in a specification meet stakeholder's needs. Correctness in AGORA is strongly related to contribution values on the path of the adopted goal as well as on its stakeholder's preference value. Particularly, measures of modifiability and desirability of goals for selection may be derived from AGORA graphs [7].

4. Architectural impact of a selection

During the "analysis" phase a COTS candidate or a set of COTS candidates is selected from several alternatives, and decisions are made based on previous definitions and measurements. Basement for decisions might include to detect sub-domains affected by the recently incorporated COTS and identify dissatisfactions – functional dissatisfactions and accuracy dissatisfactions – according to the stability state established in "definition" phase.

Architectural adaptability might define calculations for measures of changes that affect system's stability in terms of architectural adaptability. Adaptability of an architecture can be traced back to the requirements of the software system for which the architecture was developed. For example, the POMSAA (Process-Oriented Metrics for Software Architecture Adaptability) framework [13], achieves the need of tracing by adopting the NFR framework that is a process-oriented qualitative framework for representing and reasoning about NFRs (non-functional requirements). In the NFR Framework, the three tasks for adaptation become softgoals to be achieved by a design for the software system. An adaptable component of a software system should satisfice these softgoals for adaptation. We refer the reader to [13] for a more detailed description on the computation of the metrics.

5. Challenges

We have chosen some activities to exemplify some of the main tasks of a Six Sigma-based process, however several challenges remain.

Firstly, setting the right requirements for selection, and identifying domains and mappings of S_C are not easy tasks. Different scenarios should guide the process, but taking into account the different goals that are relevant in each stage. Then, goals are discovered and refined iteratively to reach commitment among all stakeholders involved in the process. For example, the existence of scenarios might be guided by the discovery process at different levels as introduced by C. Rolland et al. [14]. However, expertise and knowledge to define scenarios are here as important as they are in traditional elicitation processes.

Secondly, once requirements are categorised and weighted, a process to obtain product and vendor information should be carried out. Sources of vendor names include research services, Internet searches, networking, industry group publications and contacts, advertisements in IT journals, and so forth. Also, product information provided by vendors should be assessed. For example, the survey in [8] evaluates how much of the information required to assess COTS components is actually available from vendors' information. The main goal of the survey is to analyse the current gap between the required and provided information, so refinement of quality models can be guided to reduce the gap, yielding in more realistic attributes and metrics. Clearly, collecting effective measures is highly dependent of the amount and quality of the information provided by third-parties.

Finally, determining the impact of a component solution on a given architecture (measured in terms of architectural adaptability in our proposal) is a quite outstanding challenge. We suggest here integrating qualitative and quantitative measures for analysis considering that: (1) qualitative judgments are based on more precise values to be evaluated, which reduce ambiguity of the evaluation process; (2) quantitative values provide a more objective scale for comparison among alternatives; and (3) design decisions might be more formally specified.

6. Acknowledgments

This work is partially supported by the CyTED (Ciencia y Tecnología para el Desarrollo) project VII-J-RITOS2, the UNComa project 04/E048, and the MAS project supported by the Dirección General de Investigación of the Ministerio de Ciencia y Tecnología (TIC 2003-02737-C02-02).

7. References

[1] A. Cechich, M. Piattini, and A. Vallecillo (Eds.), *Component-Based Software Quality: Methods and Techniques*, Springer-Verlag LNCS 2693, 2003.

[2] A. Cechich and M. Piattini, "Defining Stability for Component Integration Assessment", *Proceedings of the 5th International Conference on Enterprise Information Systems*, ICEIS, Angers, France, 2003, pp. 251-256.

[3] A. Cechich and M. Piattini, "On the Measurement of COTS Functional Suitability", *Proceedings of the 3rd International Conference on COTS-based Software Systems*, ICCBSS, Springer-Verlag LNCS, California, USA, 2004.

[4] C. Alves, "COTS-Based Requirements Engineering", In *Component-Based Software Quelity: Methods and Techniques*, Springer-Verlag LNCS 2693, 2003, pp. 21-39.

[5] N. Maiden and C. Ncube, "Acquiring COTS Software Selection Requirements", *IEEE Software*, Vol. 15(2), 1998, pp. 46-56.

[6] P. Allen and S. Frost, "Planning Team Roles for CBD", In *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, 2001.

[7] A. Cechich and M. Piattini, "Balancing Stakeholder's Preferences on Measuring COTS Component Functional Suitability", *Proceedings of the 6th International Conference on Enterprise Information Systems*, ICEIS, Porto, Portugal, 2004, (to appear).

[8] B. Bertoa, J. Troya, and A. Vallecillo, "A Survey on the Quality Information Provided by Software Component Vendors", *Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, QAOOSE, Darmstad, Germany, 2003.

[9] C. Tayntor, *Six Sigma Software Development*, Auerbach Publications, 2002.

[10] A. Cechich and M. Piattini, "Managing COTS Components Using a Six Sigma-based Process", *Proceedings of the 5th International Conference on Product Focused Software Process Improvement*, PROFES, Springer-Verlag LNCS, Kansai Area, Japan, 2004, (to appear).

[11] R. Alexander and M. Blackburn. Component Assessment Using Specification-Based Analysis and Testing. *Technical Report SPC-98095-CMC, Software Productivity Consortium*, 1999.

[12] H. Kaiya, H. Horai, and M. Saeki, "AGORA: Attributed Goal-Oriented Requirements Analysis Method", *Proceedings of the IEEE International Conference on Requirements Engineering*, 2002, pp. 13-22.

[13] L. Chung and N. Subramanian. "Process-Oriented Metrics for Software Architecture Adaptability", *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, 2001, pp. 310-312.

[14] C. Rolland, C. Souveyet, and C. Ben Achour, "Guiding Goal Modelling using Scenarios", *IEEE Transactions on Software Engineering*, 24(12), 1998, pp. 1055-1071