A Six Sigma-based process to improve COTS component filtering

Article *in* Journal of Research and Practice in Information Technology · December 2007

CITATIONS
0

READS
331

2 authors:

Alejandra Cechich
National University of Comahue
158 PUBLICATIONS **1,110** CITATIONS
SEE PROFILE

Mario Piattini
University of Castilla-La Mancha
1,216 PUBLICATIONS **15,168** CITATIONS
SEE PROFILE

A Six Sigma-Based Process to Improve COTS Component Filtering

Alejandra Cechich

Departamento de Ciencias de la Computación. Universidad Nacional del Comahue
Buenos Aires 1400
8300 Neuquén (Argentina)
acechich@uncoma.edu.ar

Mario Piattini

Departamento de Informática. Universidad de Castilla-La Mancha
Paseo de la Universidad, 4
13071 Ciudad Real (Spain)
Mario.Piattini@uclm.es

Typically, COTS component evaluations embody a first stage intended to determine rapidly which products are suitable in a target context. This stage – called "filtering" or "screening" – chooses a set of alternatives to be considered for identification and more detailed evaluation. For successful filtering processes, composers increasingly focus on closing the gap between required and offered functionality, hence reducing ambiguity of information for comparison. However, it is quite difficult to establish a framework for managing unstructured information at this early stage. In this paper, we introduce filtering immersed in a Six Sigma-based process, aiming at improving the filtering process itself as well as its deliverables. We illustrate the proposal by a case study.

Key words: COTS (Commercial Off-The-Shelf) Components. COTS Component Selection. COTS-Based Project Management. Six Sigma-Based Processes.

ACM Classification: D.2.1 – D.2.9 – D.2.8 – D.2.13

1. INTRODUCTION

The acceptance of COTS-Based Software Development (CBSD) carries along several challenges to the techniques and tools that have to provide support for it. Some of them are known problems, such as dealing with Commercial Off-The-Shelf (COTS) component production and integration, reducing complexity while improving reliability of components, and creating interaction models and supporting middleware (Cechich, Piattini, and Vallecillo, 2003).

In addition to these difficulties there are other factors that complicate the job of an individual or group responsible for ensuring COTS-Based system quality. The first factor concerns uncertainty as to what constitutes the component resource quality. The answer is neither obvious nor simple. One might be tempted to reply, "quality features of a software element independently implemented and composed without modification". What about its documentation? And the degree to which a component or process meets specified requirements? To narrow the scope, one might suggest certain key quality features of a composition. But what if these features rely on components from

Manuscript received: 23 September 2006
Communicating Editor: Graham Low

unreliable, external sources? Should a composer attempt to work with those sources to improve quality? Hence, software quality staff must rethink the way software is assessed, including all life-cycle phases – from requirements to evolution.

When selecting COTS components, there is usually a quick first effort intended to determine very rapidly which products are unsuitable in the current context. This process, called *filtering*, can be initiated as soon as there is at least one relevant alternative to be considered. The initial filtering might be carried out based on different information to be collected, including vendor capability, legal aspects on the use of the component, offered functionality, etc. All these evaluations complement each other, and all of them deserve careful examination. However, functionality constitutes a main concern since the number of components in the solution must be minimized while the contribution of functionality is maximized (Abts, 2002).

Here, we should note that a related work by Bianchi, Caivano, Conradi, Jaccheri, Torchiano and Visaggio (2003), defines the quality characteristic "adequateness" of a COTS product by considering the application domain, the functionality provided by the COTS product, and its effective usage within the CBS. Authors have introduced two metrics – "Functional Coverage" and "Compliance" – to refer to the concept of functionality as broad as possible. Rationale behind our proposal is similar to those considerations; however, they do not address the problem of calculating "the number of functionality" as required by the metrics.

PORE (Procurement-Oriented Requirement Engineering) (Ncube and Maiden, 1999) selects products by rejection, i.e. products that do not meet core customer requirements are selectively and iteratively rejected and removed from the candidate list. This process shows that increasing the number and detail of requirement statements will decrease the number of COTS candidates. For requirements acquisition, PORE divides the process into stages and provides three templates for three key stages of the process (Maiden and Ncube, 1998). Particularly, the first template is used during early stages of COTS component selection, when the evaluation team deals with Web site information, technical documents, market analyses, etc., which are characterized by the lack of detailed data. One of the problems of the PORE method is that the iterative process of requirements acquisition and product evaluation/selection is very complex. At any point, a large number of possible situations can arise, or a large number of processes and techniques to use in a single situation can be recommended. To handle this scale of complexity, a prototype tool known as PORE Process Advisor has been developed.

Finally, in the Function Fit Analysis (Holmes, 2004), "fit" is defined as the amount of out-of-the-box functionality that can be used without any modifications. Using this definition, a comparison of the requirements function point count to the COTS function point count results in calculating the "fit" percentage of the COTS. However, what "functionality" means is not addressed by the proposal.

Although functionality also guides our process, the problem of ensuring component and component composition quality is exacerbated by the multiplicity of potential problems with COTS components. Whether you have built your system using COTS components from many vendors, or a single vendor has provided you with an integrated solution, many of the risks associated with system management and operation are not in your direct control (Lipson, Mead, and Moore, 2002). Particularly, selecting COTS is not all about confronting components services against requirements. For example, some methods include a supplier selection process where the objectives are to establish supplier selection criteria, evaluate suppliers, rank them according to the criteria and select the best supplier(s). Our approach focuses on measuring the component's functional suitability at early stages; however other criteria such as strategic criteria (cost factors, risk factors), or non-

functional criteria (reliability, performance, etc.) could extend the proposal. Here, we should note that including any additional criteria could complicate filtering; hence, extending criteria should deserve careful examination.

Spiral approaches define a series of iterations to mitigate risk while addressing the most critical functions (Biehl, 2004; Boehm, Bose, Horowitz and Lee, 1998; Tayntor, 2002; Tyson, Albert and Brownsword, 2003). The proposal by C. Albert *et al* (Albert and Brownsword, 2002), called Evolutionary Process for Integrating COTS-Based Systems (EPIC), builds upon of the elements of the Rational Unified Process (Jacobson, Booch and Rumbaugh, 1999) and disciplined spiral engineering practice (Boehm, 1998). Particularly interesting to filtering is the "Inception Phase", which establishes a common understanding among stakeholders of what the solution will do. It ends when it is demonstrated that one or more candidate solutions can be integrated into the organization's architecture. This phase accumulates information to identify measurable criteria that correspond to preliminary expectations; however the lack of particular measurement procedures – and metrics – makes this process perfectible.

On these lines, our approach focuses on fact-based decisions and teamwork, which might drive the identification and filtering process by using specific measures (Cechich and Piattini, 2004c; Cechich and Piattini, 2004d). Particularly, we consider functional suitability as the main aspect to be measured; however, measures should be expressed in such a way that calculation is possible at early stages.

Our main contribution is focused on the introduction of a filtering process based on improvement. To do so, we built upon the Six Sigma approach and mirrored their phases to facilitate filtering. Additionally, our process might be extended by classifying and standardizing information for analysis, building upon some work in this Field (Ayala, Botella and Franch, 2005; Bianchi, Caivano, Conradi, Jaccheri, Torchiano and Visaggio, 2003; Braga, Mattoso and Werner, 2001; Jaccheri and Torchiano, 2002; Mielnik, Lang, Laurière, Schlosser and Bouthors, 2003; Pahl, 2003; Torchiano and Morisio, 2004).

In this paper, we introduce the Six Sigma-Based process and focus on lessons learned and managerial implications for COTS component filtering. Section 2 briefly introduces our filtering process. Then, Section 3 shows our adaptation to Six Sigma and discusses specific techniques. Section 4 introduces a case study focusing on some of these techniques; and Section 5 illustrates implications. Finally, we address conclusions and future work.

2. A COTS COMPONENT FILTERING PROCESS

After an exploratory study investigating seven projects, Torchiano *et al* claim that "Formal selection procedures are seldom used. Familiarity with either product or generic architecture is the key factor in selection" (Torchiano and Morisio, 2004). So, although a COTS selection process may be as important to small projects as it is to larger ones, current processes are sometimes difficult to adopt. Then, our work was motivated by the fact that many methods and techniques for discovering and filtering COTS component candidates currently include complex criteria and qualifying thresholds for selection, and particularly for filtering. Our model was built after analysing different aspects of component filtering, receiving feedback from scholars and practitioners in the Field. After several attempts to improve the process, we arrived to the one introduced in this Section.

To clarify the process's view, the following diagrams summarize it by using the notation introduced by Marca and McGowan (1988).

Figure 1 shows the external interfaces for the "Filtering" process (Cechich and Piattini, 2005a). The *quality thresholds/organizational constraints* control consists of attributes that influence or

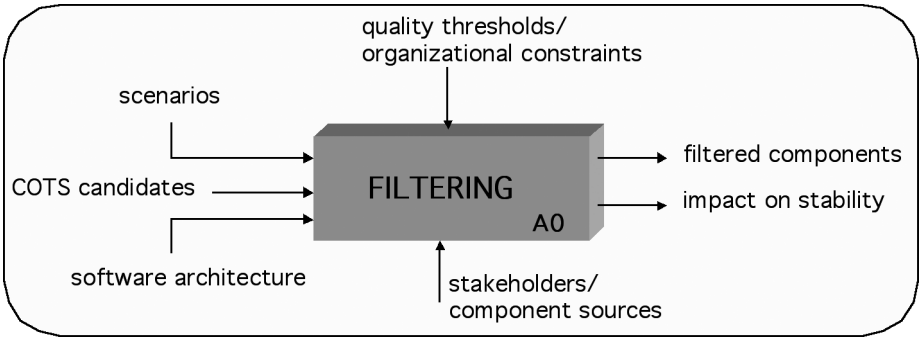

Figure 1: Contextual view of the filtering process

constrain systems' requirements and the filtering process. Typically, the constraint scope will include aspects such as schedule, cost, context and domain. Quality thresholds represent the acceptance thresholds associated to quality attributes of the system.

The *scenarios* input consists of different sequences of behavior depending on the particular requests made and conditions surrounding the requests; the *COTS candidates* input consists of a number of COTS components available from marketplace; and the *software architecture* input consists of the architectural basic units, components, and relationships among them. At this level, a basic unit for architecture is undetermined allowing multiple instantiations – such as compound units, corresponding processes, etc.

The *stakeholders/component sources* mechanism consists of people who are involved in the filtering process. *Component sources* represent the external resources that are explored in the search of COTS candidates.

The *impact on stability* output consists of an identification of functional dissatisfactions according to the stability state of the architecture. This output might include suggestions for new requirements or requirement updates discovered during the search for COTS candidates as well as suggestions for architectural changes (Cechich and Piattini, 2003). Finally, the *filtered components* output consists of the component or components chosen for more evaluation as a result of the filtering process.

There are three primary activities in the Filtering Process: a commitment process, a pre-filtering process, and a final filtering process, as shown in Figure 2.

The Commitment process in the decomposition contains the following activities (as shown in Figure 3):

- "Derive Goals" determines the stability status of the system and provides a component specification to be committed. This activity uses information from scenario and software architecture specifications. *Stakeholders* use scenario authoring and goal discovering to elicit requirements at different levels of detail, and an abstract *component specification* is provided as input to the "Compute Preferences" process. *Desirability* is used to iteratively calibrate the abstract specification until a *committed specification* is produced as output. The *goals* output consists of goals to be refined and weighted during the "Compute Preferences" process. Information of functionality and adaptability is used as a refinement constraint, i.e. they drive the activities helping decide on further searching and evaluation of candidates.
- "Compute Preferences" calculates preference indicators such as desirability and/or modifiability, and stakeholders' preferences on refined goals (Cechich and Piattini, 2004b).

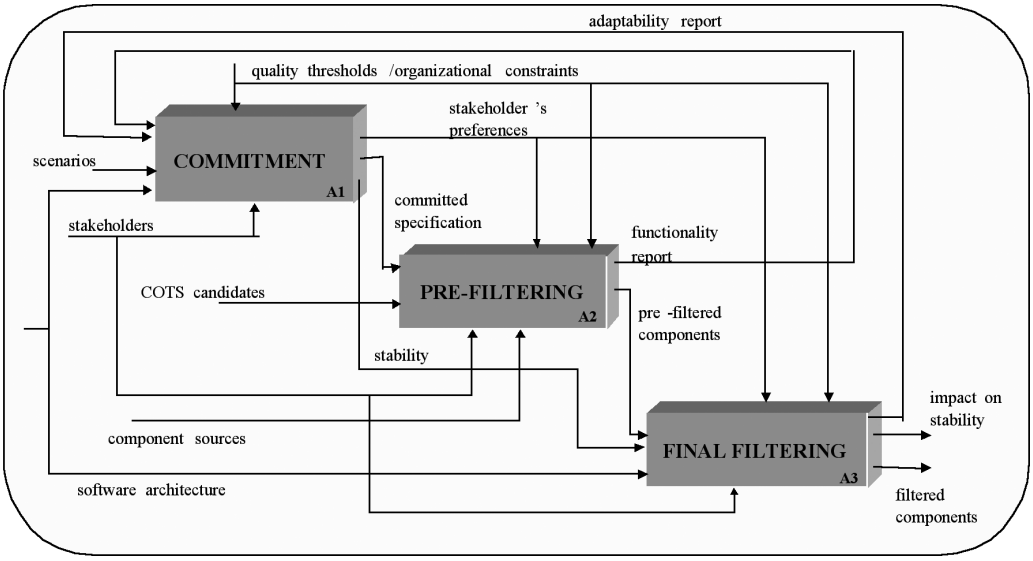

Figure 2: Primary activities of the filtering process

The Pre-Filtering process in the decomposition contains the following activities (as shown in Figure 4):

- "Functional Suitability Measurement" computes metrics on functional suitability of COTS candidates (Cechich and Piattini, 2004c). *Component sources* are used as a mechanism to search candidates. Then, COTS candidates from a marketplace are chosen and *Functional suitability metrics* are produced as input to the "Functional Suitability Analysis" process. The *committed specification* acts as a guideline to *stakeholders*, who drive the search procedure. Information from functionality is used as a refinement constraint similarly to other activities in the process.

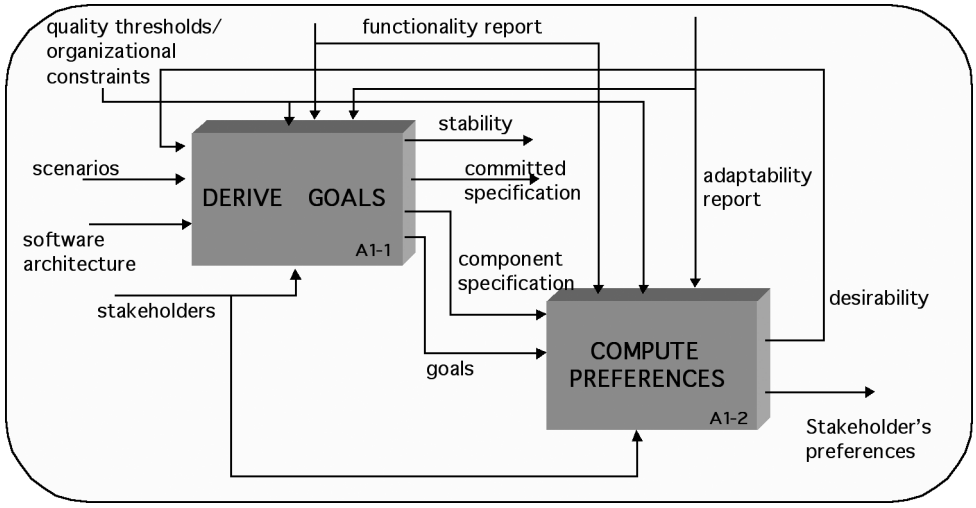

Figure 3: Commitment process

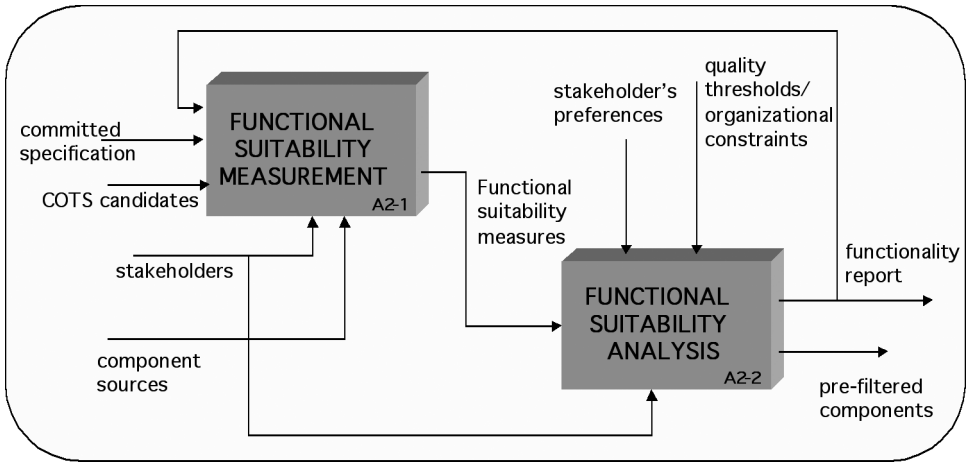

Figure 4: Pre-filtering process

- "Functional Suitability Analysis" analyses metrics on functional suitability measured for COTS candidates. A *functionality report* summarizes the results from the analysis and serves as an indicator to decide on how to stop the search. Modifiability constrains the analysis taking into account the degree in which goals can be modified.
- The *pre-filtered components* output consists of the component or components that are functionally suitable, and hence candidates for further evaluation.

The Final Filtering process in the decomposition contains the following activities (as shown in Figure 5):

- "Architectural Adaptability Measurement" computes metrics on architectural adaptability (size and complexity of adaptation, and semantic architectural adaptability) on the given *software architecture* and considering a set of *pre-filtered components* (Cechich and Piattini, 2004d). Then, *architectural adaptability metrics* are produced as input to the "Architectural Adaptability

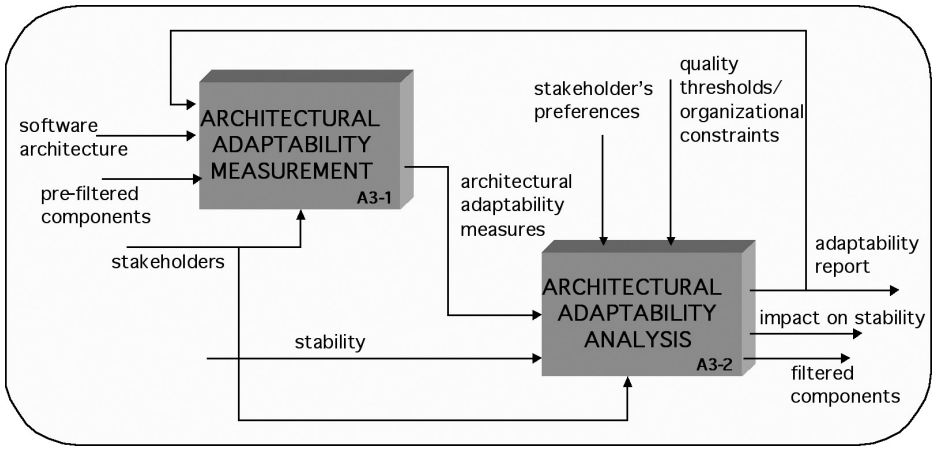

Figure 5: Final filtering process

Analysis" process. The *software architecture* acts as a basis for judgments of *stakeholders*. Information from adaptability is used as a refinement constraint similarly to other activities in the process.

- "Architectural Adaptability Analysis" analyses metrics on architectural adaptability. An *adaptability report* summarizes the results from the analysis and serves as an indicator to decide on reviewing stakeholder's judgments and/or continuing the search for candidates. The *filtered components* output consists of the component or components that are finally filtered. The *impact on stability* output reports on the degree in which initial system's stability is affected by the filtered components.

3. ADAPTING THE FILTERING PROCESS TO SIX SIGMA

Six Sigma is an approach to product and process improvement that has gained wide acceptance and has delivered large business benefits across many industries (Biehl, 2004; De Feo and Bar-El, 2002; Gack and Robinson, 2003; Tayntor, 2002). While the objectives of Six Sigma are to reduce variation and prevent defects, it is also a management philosophy that includes the need for fact-based decisions, customer focus, and teamwork. Six-Sigma precepts can guide stakeholders through the pre-evaluation of COTS components and help increase the probability of success. In Tayntor (2002), the Six Sigma approach has been suggested selecting packaged software, however the evaluation mainly relies on the information provided by demos and additional documentation of the COTS software. Then, the lack of measures makes this process perfectible.

Six Sigma is typically divided into five phases, creating what is referred to as DMAIC, which is an acronym for the following phases:

1. *Define* the problem and identify what is important: Identify the problem and the customers; define and prioritize the customer's requirements; and define the current process.
2. *Measure* the current process: Confirm and quantify the problem; measure the various steps in the current process; revise and clarify the problem statement, if necessary; and define desired outcome.
3. *Analyze* what is wrong and potential solutions: Determine the root cause of the problem; and propose solutions.
4. *Improve* the process by implementing solutions: Prioritize solutions; and develop and implement highest benefit solutions.
5. *Control* the improved process by ensuring that the changes are sustained: Measure the improvements; communicate and celebrate successes; and ensure that process improvements are sustained.

Our approach defines the five-phases of the Six Sigma-Based process from the activities depicted in the previous section. To clarify this point, let us consider the following relationships.

- *What should I search?* ("Define-Commitment"). In this scenario, the team simplifies requirements to get functional goals, for example, specified as use cases. Search will be limited to COTS candidates that prove to be fully functional compatible with respect to these specifications. So, functional partitioning and decomposition generate what we should search in traditional way. Participants of the process express their commitment with the required functionality, thus a committed specification becomes the filtering starting point.
- *How should I pick one?* ("Measure/Analyze"-"Pre-filtering"). In this scenario, the team applies functional suitability measures to get suitable candidates to be subjected to further evaluation. A

high-level definition allows us to adapt and calculate the values. So, candidates are pre-selected according to the resulting values.

- *How should I pick one based on later integration?* ("Measure/Analyze"-"Final-Filtering") Here, compliance with standards and other architectural issues are broadly analysed during filtering; allowing the team to suggest changes on designs as well as evaluate other possibilities. It would lead to reducing risks of selection by analysing incompatibilities at early stages.
- *What should I know?* ("Improve"-"Filtering") Here, we assume the team knows well-known software engineering methods and techniques, such as use case modelling; and learns some others, such as goal prioritisation through goal-oriented graphs. Some of the techniques are considered optional during the process, and their application depends on the context, which is clarified before starting. So, learning is reduced by simplifying the search – at the beginning limited to only functional concerns – and by applying simpler procedures.
- *When should I stop?* ("Improve/Control"-"Filtering"). We have already mentioned that the results from the functional and/or architectural adaptability analysis serve as an indicator to decide on how to stop the search. Clearly, as Figure 6 shows, architectural features and software requirements (scenarios) are the main inputs to drive our search of COTS candidates. From the architectural point of view, there are some additional remarks. Firstly, the impact on stability is currently based on qualitative judgements on semantic architectural adaptability, although they are combined with quantitative values of complexity and size of adaptation. We suggest here that quantitative and qualitative metrics together would help reach agreement when a decision

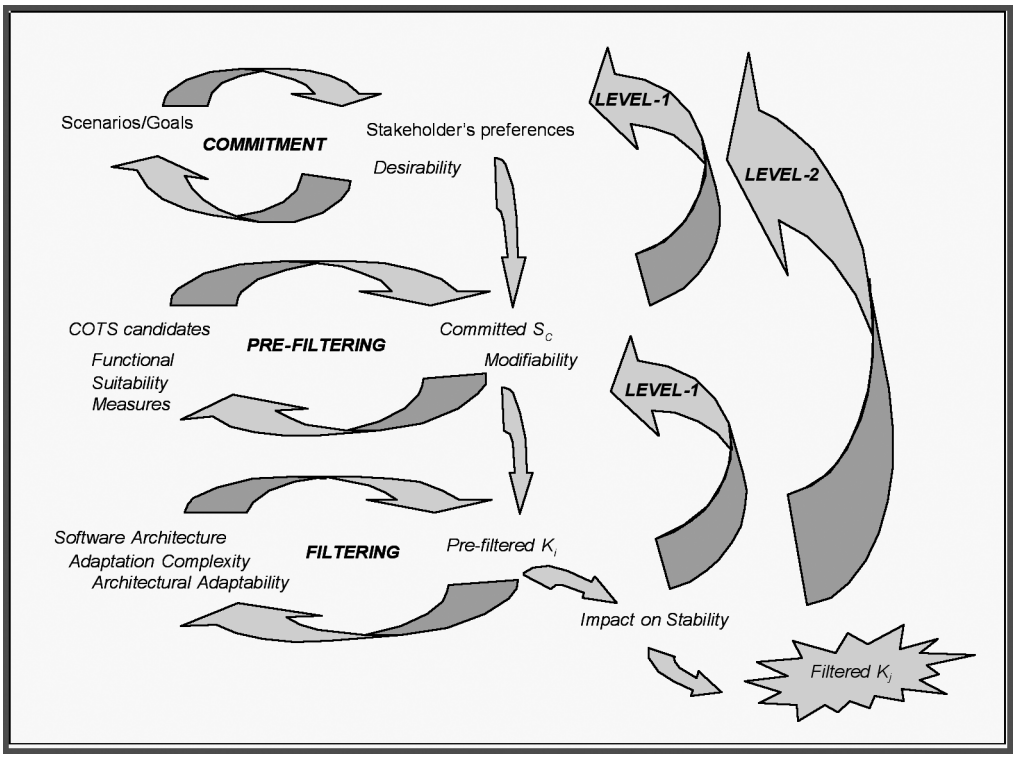

Figure 6: The iterative nature of our filtering process

on filtering components must be made. However, this agreement is not reached instantaneously. Secondly, basis for decisions includes detecting architectural artefacts affected by the COTS candidate and identifying functional dissatisfactions. Causes of dissatisfaction should drive the process possibly introducing changes on the requirements definition, the host architecture, and even on the filtering process itself. Therefore, our highly iterative process essentially will stop when expectations are reached through a COTS solution; or when the effort of searching for more candidates becomes too costly. Of course, decisions made during the process might be weighted, in such a way that different roles and stakeholders' expertise are explicitly incorporated. Another concern is on actually the number of possible candidates on a marketplace. Is the searched functionality common enough to be offered by the marketplace? How easy is finding COTS components actually? Obviously, considerations on when to stop filtering might be traded-off by considering several other factors.

Additionally, in Figure 7 deliverables are summarized to indicate the existence of documentation composed of criteria for selection and measures that identify suitability with respect to a source system. Tools in the Figure refer to well-known tools for quality assessment, as well as specific tools and measures we proposed for filtering COTS components.

3.1 The Define Phase

To ensure that decisions are fact based, it is important that the "define" phase be reinforced. Although it is always important to understand the current process and the problem to be solved, the

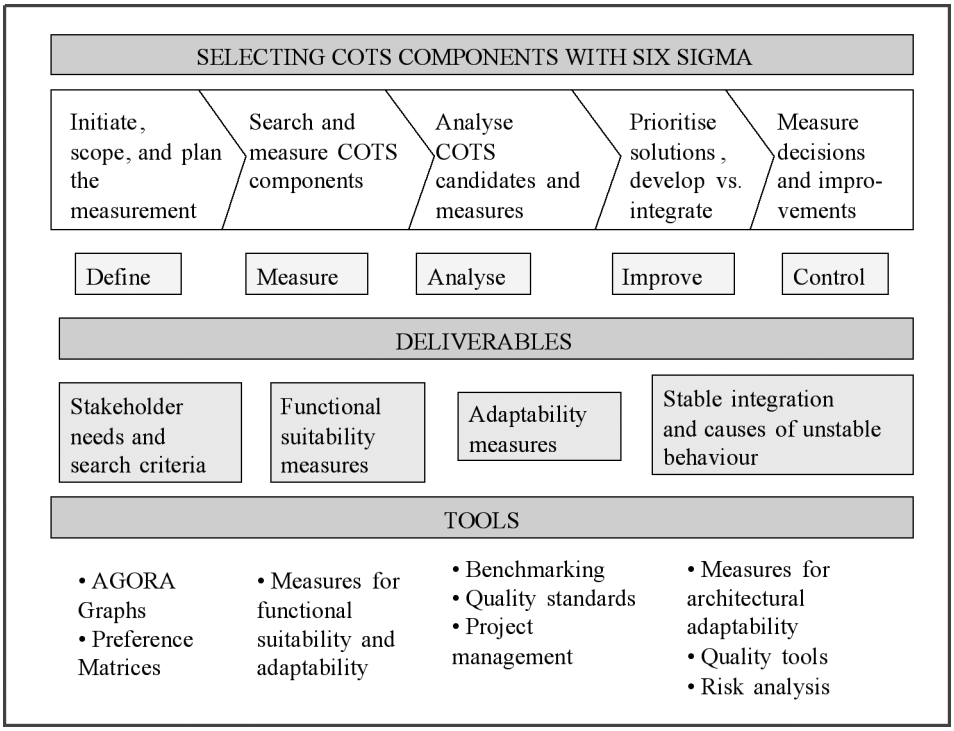

Figure 7: A Six Sigma-based filtering process

importance is greater when filtering COTS components because requirements and services must be balanced as part of a negotiation procedure.

Firstly, COTS component's required functionality should be expressed to define search goals and criteria for evaluation. It will produce the first deliverable in Figure 7 called "Stakeholders needs and search criteria". To do so, we have adapted the model in Alexander and Blackburn (1999), 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 evaluating a component from a set of candidates.

According to Alexander and Blackburn, a system can be extended or instantiated through the use of some component type. Because 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* defines a specification Sc for the abstract component type C. Any component Ki, that is a concrete instance of C, must conform to the interface and behavior specified by Sc, as shown in Figure 8 (from Alexander and Blackburn, 1999).

Identifying domains and mappings of Sc is not an easy task. 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 (Cechich and Piattini, 2004b).

Let us clarify this point. Suppose we have discovered the relevant goals to be achieved by components that instantiate the required specification of Sc. But when incorporating COTS components, goals should be balanced against COTS services. Different sequences of behaviour, or scenarios can unfold, depending on the particular requests made and conditions surrounding the requests. The use case collects together those different scenarios (Cokburn, 2001).

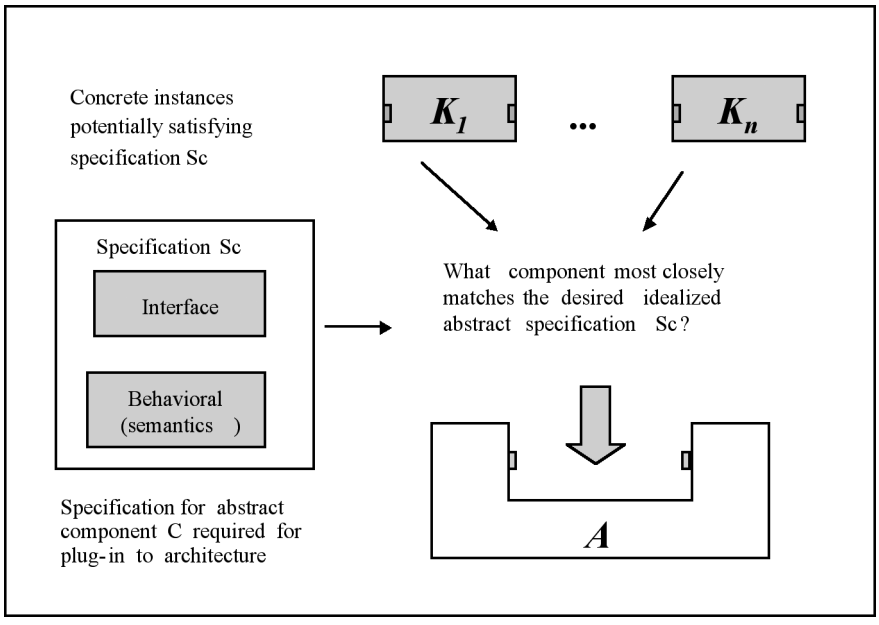

Figure 8: A required specification Sc for the component C

Therefore, having discovered relevant goals from scenarios and with the functional goals of the component specified by mappings in Sc, the next step is to refine the goals considering the perspectives of different stakeholders. For example, a reuse architect may be interested in identifying and acquiring components promoting the value of reuse and ensuring consistency of design across projects; or a certifier may be interested in setting component specification standards and ensuring compliance and consistency of components across different teams (Allen and Frost, 2001). Hence, functional requirements are affected by different views that should be conciliated.

Then, the computation of stakeholders' preference values for the refined goals will allow us to add preferences to mappings of Sc. To do so, our proposal extends the version of a Goal-Oriented Requirements Analysis Method, called AGORA (Kaiya, Horai and Saeki, 2002; Kaiya and Saeki, 2004), which is a top-down approach for refining and decomposing the needs of customers into more concrete goals that should be achieved for satisfying the customer's needs.

In our context of COTS-based systems, an AGORA graph describes the required specification of a required component (Sc) according to the scenario S. 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 stakeholders' needs. Correctness in AGORA is strongly related to contribution values on the path of the adopted goal as well as on its stakeholders' preference value. It allowed us to define and calculate two interesting indicators for filtering – *desirability and modifiability*.

Desirability might reduce search and evaluation efforts by detecting functionality on which there is no enough agreement; and modifiability might help to predict a space of negotiation and change when constraints from actual candidates are applied. For specific calculations of these indicators, we refer the reader to Cechich and Piattini (2004b).

3.2 The Measure Phase

Once committed goals have been achieved, we may proceed with the filtering process by applying different measures on COTS component candidates. Firstly, we produce the "Functional suitability measures" deliverable in Figure 7 as follows. The metrics are defined considering that only high-level descriptions of the COTS components are available, without further details on data and control execution. This assumption is supported by fundamentals of the filtering process, and by the characteristics of information provided by most components' suppliers (Bertoa, Troya, and Vallecillo, 2003).

For the measure definitions, we assume a conceptual model with universe of scenarios S, an abstract specification of a component C, a set of components K relevant to C and called candidate solution SO, and a set of pre-selected components from SO, called solution SN. Sc(i) represents the map associated to the input value i defined in the domain of Sc. This map should provide a valid value on the output domain of Sc, i.e. there is no empty maps or invalid associations. A similar assumption is made on the mappings of SK.

Then, the solution in which all components potentially contribute with some functionality to get the requirements of C is called here SN. Table 1 lists our suite of functional suitability measures, which are classified into two groups: component-level measures and solution-level measures. For example, CF_C measures the number of functional mappings provided by Sk and required by Sc in the scenario S; SN_{CF} measures the number of components that contribute with compatible functionality to get the requirements of Sc in the scenario S; and so forth. We refer the reader to Cechich and Piattini (2004c) for a more formal definition of the metrics.

Measure Id.	Description
Component-Level	
CF_C Compatible Functionality	The number of functional mappings provided by S_K and required by S_C in the scenario *S*
MF_C Missed Functionality	The number of functional mappings required by S_C and NOT provided by S_K in the scenario *S*.
AF_C Added Functionality	The number of functional mappings NOT required by S_C and provided by S_K in the scenario *S*.
CC_F Component Contribution	Percentage in which a component contributes to get the functionality required by S_C in the scenario *S*.
Solution-Level	
SN_{CF} Candidate Solution	The number of components that contribute with compatible functionality to get the requirements of S_C in the scenario *S*.
CF_S Compatible Functionality	The number of functional mappings provided by *SN* and required by S_C in the scenario *S*.
MFS Missed Functionality	The number of functional mappings required by S_C in the scenario *S* and NOT provided by *SN*.
AF_S Added Functionality	The number of functional mappings NOT required by S_C in the scenario *S* and provided by *SN*.
SC_F Solution Contribution	Percentage in which a solution contributes to get the functionality required by S_C in the scenario *S*.

Table 1: Functional Suitability Measures

There are another types of analysis the component should be exposed before being eligible as a solution (Cechich and Piattini, 2004d; Davis, Gamble and Payton; 2002). Until now, our set of measures has only provided a way of identifying suitable components from a functional point of view. In defining the final filtering phase, a composer should contrast COTS candidates against a host software architecture, and analyze architectural adaptability on the light of different stakeholders' views.

Secondly, we produce the "Adaptability measures" deliverable in Figure 7 as follows. Adaptability of an architecture can be traced back to the requirements of the software system for which the architecture was developed. The POMSAA (Process-Oriented Metrics for Software Architecture Adaptability) framework (Chung and Subramanian, 2001), achieves the need of tracing by adopting the NFR framework (Chung, Nixon, Yu and Mylopoulos, 2000) 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 satisfy these softgoals for adaptation. One of the softgoals to be decomposed is adaptability, which can be further described in terms of semantic adaptability, syntactic adaptability, contextual adaptability and quality adaptation. We suggested here to combine qualitative and quantitative metrics to measure architectural adaptability during filtering (Cechich and Piattini, 2004d).

3.3 The Analyze Phase

During the second activity of our pre-filtering phase ("Functional Suitability Analysis"), we analyze results from functional suitability measures. Among others, specification of scenarios, discovery of goals, quality of specifications, and reliability of measures, are analyzed. For example, information supplied by third-parties might not be detailed enough to identify required functionality, so a decision should be made on discharging those candidates, or reviewing and changing the required specifications. The case study in Section 4 further illustrates this point.

During the second activity of our final filtering phase ("Architectural Adaptability Analysis"), we analyze results on semantic architectural adaptability. Then, the suitable components should be analyzed taking into account all features detailed on the adaptability decomposition. Later on, other characteristics such as security, and interoperability are analyzed, and impact on stability is determined. To do so, we disaggregate the Architectural Semantic Adaptation goals and we analyze metrics from the POMSAA framework.

3.4 Improve and Control Phases

These phases are concerned with ensuring that the selected candidates can be integrated and supported within the required quality; and we produce here the last deliverable "Stable integration and causes of unstable behaviour". For example, causes of low architectural adaptability should be determined, along with causes of functional dissatisfaction. Evaluation of results can reveal problems that might arise during filtering leading to prioritize solutions deciding on pre-selecting the COTS components or building the required functionality from scratch. Different architecture's uses may necessitate changes to the same components or connections. In such a case, during filtering we should detect uses that affect the architecture keeping designs with the fewest use conflicts.

A successful COTS evaluation procedure should communicate its practices and suggests possible corrective/adaptive actions to sustain its success. Although these activities do not differ from a traditional system development project or, indeed, from a classic Six Sigma project, they are important and should not be neglected.

4. A MOTIVATING CASE STUDY: E-PAYMENT BY CREDIT CARD

Let us introduce a motivating case study. For Credit Card Payments, a service provider intermediates in credit card transaction processing. It implies that the service provider validates credit card numbers and expiration dates, obtains authorization from the credit card issuers and issues confirmation numbers to taxpayers at the end of the payment transaction. For the sake of this case, we suppose that the required functionality will be supplied by COTS components in a marketplace. So, the next step is to *define functional mappings* to proceed with the filtering process.

However, according to Yin (1994), external conditions may influence a case remarkably, which means that especially in conducting multiple-case studies external variables may make replication and cross-analysis very difficult. In this study, we have compiled different COTS candidates and analyzed the interfaces, i.e. the functional requirements that a composer has in relation to the market under study were translated into functional mappings. We should note here that the case perceived by a composer is influenced by several factors, shown in Figure 9, which affect calculation. In a particular case, adopting the process will need to focus on those factors and further detail them. As we can see from Figure 9, perception of a composer will be affected by his own capability and experience in searching and using COTS components; the complexity of the particular domain application; the granularity of the component he is looking for; organizational features, such as

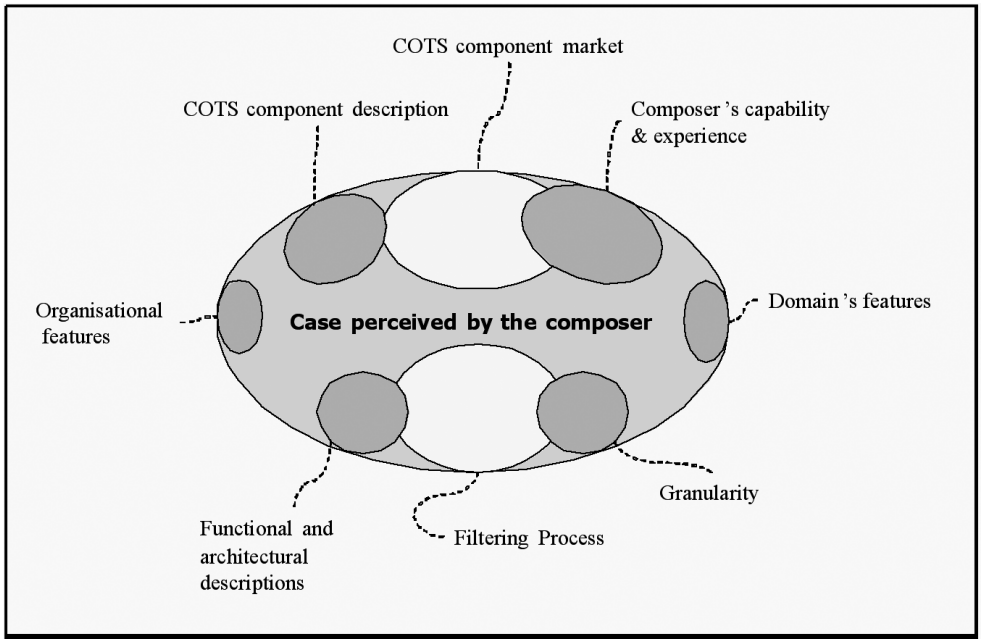

Figure 9: Influences on the filtering case

institutionalized practices for development and assessment; the description adopted for specifying functional and architectural features of the target application; the description supplied by vendors; the availability of products from a COTS component market; and the filtering process itself. All these factors are instantiated during a particular filtering letting composers adapt it to different levels of COTS-based development maturity. In this way, any organization may use the filtering process by only choosing the adequate practices for its level. Of course, the organization should also suggest improvements to those practices, as our Six Sigma-based approach defines.

Our study is about early detection of COTS candidates, but it is important to emphasize that the purpose is not to make investigations to find out things such as what supplier companies exist, what kinds of products are available etc. This kind of work would call for market research in a more practical sense, and scanning of the market is something that companies need to do. Rather, in our case the phenomenon under study is the market that a specific composer perceives.

In our case study, we want to analyze a specific case in order to learn more about other cases as well. Thus, a certain amount of typicality is needed in selecting the case. However, determining the criteria for typicality may be, of course, a complex task. In this, the most important guideline is the purpose of the study, which should form the basis for determining the case selection criteria. In addition to this, other very important issues influencing the selection are factors such as time limitations and access.

On the basis of the discussed criteria for finding the case, we will focus on the E-payment by credit card case mentioned above. Electronic payment accomplishes the criteria afore mentioned since there are more than enough COTS candidates on the market; knowledge is broad enough to be easily understood; and functionality is split into authorization and capture, which in turn imply the existence of several functions.

4.1 The Define Phase

Generally speaking, "Authorization" and "Capture" are the two main stages in the processing of a card payment over the Internet. Authorization is the process of checking the customer's credit card. If the request is accepted, the customer's card limit is reduced temporarily by the amount of the transaction. Capture is when the card is actually debited. This may take place simultaneously with the authorization request.

From these scenarios, we might produce a simplified abstract specification of the input and output domains of Sc as follows[1]:

- Input domain:
  - (AID) Auth IData {#Card, Cardholder_Name, Exp_Date, Bank_Acc, Amount} ;
  - (CID) Capture IData {Bank_Acc, Amount} .
- Output domain:
  - (AOD) Auth OData {ok_Auth} ;
  - (COD) Capture OData {ok_capture, DB_update} .
- Mapping: {AID Æ AOD; CID Æ COD} .

Then, conflicts on these mappings were analyzed to detect desirability and modifiability as introduced in Section 3.1 (Cechich and Piattini, 2004b). After reaching commitment, defining mappings involved two sub-tasks – compiling COTS component information, and translating it into mappings. To do so, our case study surveyed all COTS components catalogued as members of the "Credit Card Authorization" group by the ComponentSource organization[2]. For example, we chose one component *AcceptOnline* by *Bahs Software*[3], as a candidate to provide the required functionality. To clarify the process of translation into mappings, the following section illustrate this case.

4.1.1 The AcceptOnline Case

AcceptOnline is a COM object that provides credit card processing functionality for developers. It connects to credit card processor server through the Internet via SSL protocol which guarantees secure data transfer. To minimize fraud transactions Address Verification Service(AVS) is supported. AcceptOnline COM object provides the ability to accept credit or debit cards from Website or Internet-enabled applications. Credit card transactions are processed through the secure Internet connection (with aid of SSL protocol). As credit card processor the ECHO is used. All major credit cards are supported. To accept credit cards from a customer we should get merchant account from credit card processor. AcceptOnline uses ECHO 2 as a credit card processing organization.

Properties of AcceptOnline are grouped into the following classes: merchant fields, transaction fields, and response fields. From those classes, we identify:

- Transaction_type: This field identifies the type of transaction being submitted. Valid transaction types are:
  "CK" (System check),
  "AD" (Address Verification),

[1] Mappings were identified by discovering and refining goals from scenarios according to Rolland, Souveyet, and Ben Achour (1998).

[2] www.componentsource.org

[3] Bahs Software. AcceptOnline - ActiveX DLL - V1.0 . Available at http://www.componentsource.com/ - Access October 2004.

"AS"(Authorization),
"ES" (Authorization and Deposit),
"EV" (Authorization and Deposit with Address Verification),
"AV" (Authorization with Address Verification),
"DS" (Deposit), and
"CR" (Credit).

- cc_number: The credit card number to which this transaction will be charged.
- cc_ exp_month and cc_exp_year: The numeric month (01-12) and the year (formatted as either YY or CCYY) in which this credit card expires.
- billing_phone: The shopper's telephone number.
- grand_total: The total amount of the transaction.
- merchant_email: This is the Email address of the merchant.
- order_ type: This field determines which fields are used to validate the merchant and/or hosting merchant.
- transactionStatus: Transaction Status. Valid values are: G - Approved, D -Declined, C - Cancelled, T - Timeout waiting for host response, R - Received.

Methods of AcceptOnline are specified in terms of their main focus and required input. Particularly, the SendPacket method is used to send the transaction info to the ECHOOnline server, and required properties should be filled as shown in Table 2 (requirements for CR are partially listed).

From the AcceptOnline (AOnline) description above, we might derive the following mappings related to our authorization (AS) and capture (DS) required functionality:

Input domain:
(AOnline.ASI) {billing_phone, cc_number, cc_exp_month, cc_exp_year, counter, debug, grand_total, merchant_email} ;
(AOnline.DSI) {authorization, cc_number, cc_exp_month, cc_exp_year, counter, debug, grand_total, merchant_email}

Field	CK	AD	AS	ES	EV	AV	DS	CR
authorization							Y	
billing_address1		Y			Y	Y		
billing_address2								
billing_zip		Y			Y	Y		
billing_phone		Y	Y		Y	Y		
cc_number		Y	Y	Y	Y	Y	Y	Y
cc_exp_month		Y	Y	Y	Y	Y	Y	Y
cc_exp_year		Y	Y	Y	Y	Y	Y	Y
counter			Y	Y	Y	Y	Y	Y
debug		Y	Y	Y	Y	Y	Y	Y
grand_total			Y	Y	Y	Y	Y	Y
merchant_email		Y	Y	Y	Y	Y	Y	Y
order_number							Y	Y
...								...

Table 2: AcceptOnline requirements for transactions

Output domain:
(AOnline.ASO) {TransactionStatus} ;
(AOnline-DSO) {TransactionStatus}.

Mapping:
{AOnline:ASI Æ AOnline:ASO} ;
{AOnline:DSI Æ AOnline:DSO}

After comparing AID, CID (from specification Sc) to AOnline.ASI and AOnline.DSI, we can establish the following correspondences:

AID vs. AOnline.ASI
Cardholder_name → billing_phone
#Card → cc_number
Exp_Date → cc_exp_month, cc_exp_year
Bank_Acc → merchant_email
Amount → grand_total

CID vs. AOnline.DSI
Bank_Acc → authorization and data from AOnline.ASI
Amount → grand_ total

We should note that values of the domains do not exactly match: billing_phone is used instead of Cardholder_name to identify cardholders; and merchant_email is used for Bank_Acc. Similarly, ok_Auth, ok_Capture, and BD_Update might correspond to the different values of TransactionStatus.

However, matching is possible since purpose is similar. Then, similarity is basically determined by analyzing semantics of concepts with respect to their use. This aspect introduces one of the key points that will be further discussed later. Just remember that some knowledge is needed to perform the matching, and consequently, calculate metrics.

4.2 The Measure Phase

From 22 components surveyed in October 2004, we considered 12 for pre-filtering since the other 10 components only differed in terms of their implementations, preserving the same functionality. Results of our calculations are shown in Table 3. For example, consider the AcceptOnline case again. Computing measures produces the following results, CF_C(AcceptOnline) = 2; MF_C(AcceptOnline) = 0; AF_C(AcceptOnline) = 4; and CC_F(AcceptOnline) = 1.

Note that four components provide the functionality required by our scenario. This fact would indicate that those components are pre-selected for more evaluation, since they are 100% functionally suitable (CC_F=1). For a detailed functional suitability calculations of this case study, see Cechich and Piattini (2006).

To move into the architectural adaptability measurement – and the final filtering phase – Figure 10 shows our host architecture, whose components should attend the main functionalities of an electronic payment over Internet. Payment authorization has been further decomposed to allow us to analyze architectural concerns on the pre-filtered COTS component candidates.

The three-layered architecture is composed of (1) a service interface layer, which sits on top of a Web server's basic and handles user interactions; (2) a service implementation layer, which

Component	CF_C	MF_C	AF_C	$C C_F$
AcceptOnline	2	0	4	**1**
CCProcessing	2	0	2	**1**
CCValidate	0	2	0	0
CreditCardPack	0	2	0	0
EnergyCreditCard	0	2	0	0
IBiz	2	0	2	**1**
InaCardCheck	0	2	0	0
IPWorks	2	0	1	**1**
LuhnCheck	0	2	0	0
PaymentCardAssist	0	2	4	0
SafeCard	0	2	0	0
ComponentOneStudio	0	2	**	0

Table 3: Measurement results for components in the "Credit Card Authorization" category

identifies appropriate credit card payment for the merchant; and (3) a service-related data layer, which manages access to databases and repositories.

The component (or components) in charge of implementing credit card payment should allow us to process information for the most commonly used credit cards (Visa, MasterCard, etc.) as well

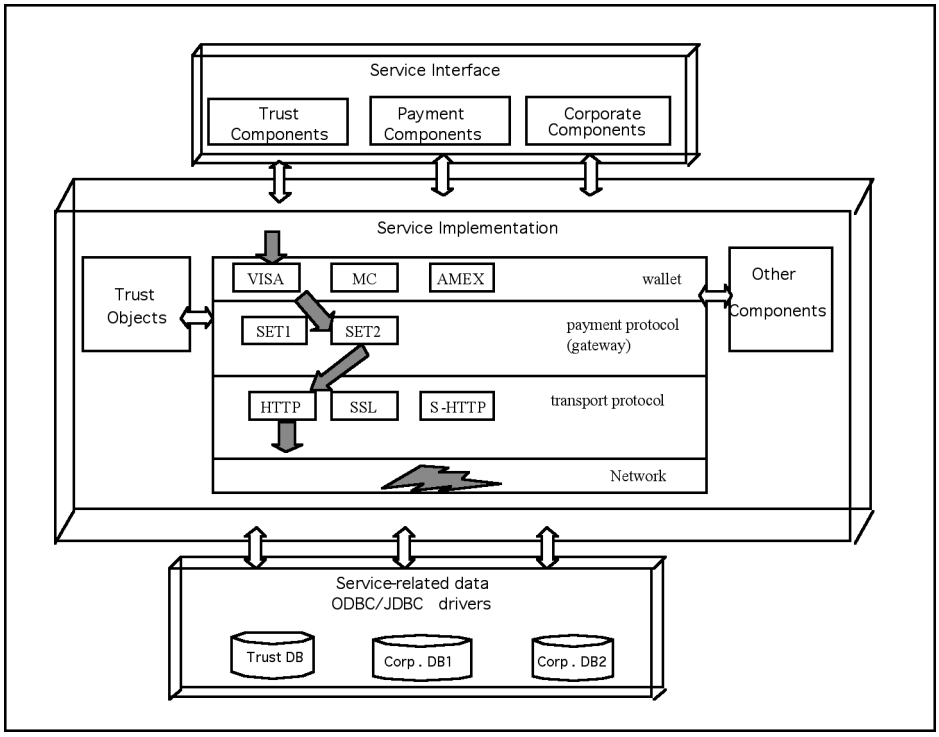

Figure 10: Host architecture for E-payment by credit card

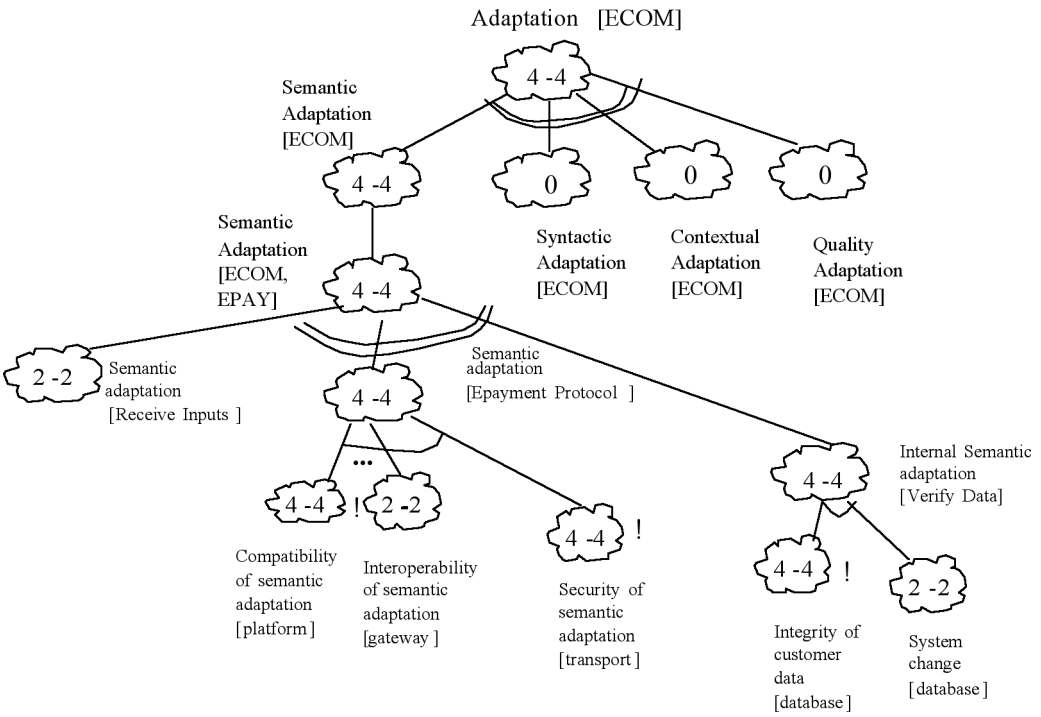

Figure 11: Architectural adaptability features to evaluate COTS candidates

as defining the possible gateways with which transactions will be processed. All transactions should be accomplished through a secure HTTPS Post to any supported gateway. Finally, the Java-based host architecture constrains components to JavaBeans from the marketplace.

By applying the POMSAA framework, we calculated a qualitative measure based on those architectural features and requirements of adaptability. Original adaptation is OR-decomposed (shown by the double arc in Figure 11) into the four subgoals – Syntactic adaptation[ECOM], Semantic adaptation[ECOM], Contextual adaptation[ECOM], and Quality Adaptation[ECOM].

Since we are interested in early detection of architectural concerns, Figure 11 shows a hierarchical decomposition of semantic adaptation. The semantic adaptation of ECOM can occur in one of three ways – due to the input receiving layer, which was measured during the pre-filtering activity; the electronic payment component or the data handling layer. This translates into an OR-decomposition of the softgoal Semantic Adaptation[ECOM,EPAY] into the three subgoals Semantic Adaptation[Receive Inputs], Semantic Adaptation[Epayment Protocol], and Internal Semantic Adaptation[Verify Data].

Semantic Adaptation[Receive Inputs] is not further decomposed due to its main assessment concerns the pre-filtering stage of our process.

Internal Semantic Adaptation[Verify Data] is delegated to a lower layer of the architecture, whose functionality should not necessarily be supplied by COTS components.

Finally, in our case, semantic adaptability of candidates was analyzed in terms of compatibility (platform-related concerns); interoperability (gateway-related concerns); and security (secure transport and processing). Those features are applied to get the three adaptation characteristics as

leaf softgoals for Semantic Adaptation[Epayment Protocol]. It is AND-decomposed into Compatibility of Semantic Adaptation[platform], Interoperability of Semantic Adaptation[gateway], and Security of Semantic Adaptation[transport].

The values for metrics that we will be coming up with are one of the many possible schemes for translating preferences into numbers. Figure 11 also gives the maximum and minimum values (the numbers inside the clouds) for the metrics of different softgoals based on the number scheme (for a detailed description of the POMSAA algorithms, (Chung and Subramanian, 2001)).

Then, for the four COTS candidates that have passed our pre-filtering process – AcceptOnline, CCProcessing, IPWorks, and IBiz – we gathered information to characterize the features we considered relevant and calculated metrics on architectural adaptability as defined previously.

4.3 The Analyze Phase

When analyzing functional suitability, note that our use of scenarios is a brief description of some anticipated or desired use of a system. In some cases, this diversity of concerns produces fine-grained functionality described by scenarios, but coarse-grained functionality might be described as well.

Let us look at the AcceptOnline case one more time. These results indicate that the AcceptOnline component has proved being 100% ($CC_F = 1$) functionally suitable, and thus a candidate for further evaluation during the filtering process. Measures also indicate that there are four added functions ($AF_C = 4$), which deserve more careful examination. However, after a closer look at those functions, we realized that many of them allow for possible variations of the authorization and capture functionalities – by considering "Address Verification" as complementary to other functions (for example, "Authorization with Address Verification"). On the other hand, "Credit" is actually adding functionality, since this function reimburses payments to the cardholder. This function has not been considered as relevant to our case because credit card payments cannot be cancelled. Taxpayers can call the credit card issuer or credit card payment service provider's customer service number to report problems such as unauthorized charges or concerns regarding payment errors. Finally, the function "CK" checks the system – a useful supporting function but not domain-oriented, and therefore not relevant to our case.

Component	CF_C	MF_C	AF_C	CC_F
AcceptOnline	3	0	1	**1**
CCProcessing	3	0	1	**1**
CCValidate	1	2	0	0.33
CreditCardPack	1	2	0	0.33
EnergyCreditCard	1	2	0	0.33
IBiz	3	0	1	**1**
InaCardCheck	1	2	0	0.33
IPWorks	3	0	0	**1**
LuhnCheck	1	2	0	0.33
PaymentCardAssist	1	2	4	0.33
SafeCard	1	2	0	0.33
ComponentOneStudio	0	3	***	0

Table 4: Measurement results after changing scenarios

Table 4 shows our measures considering the assumptions: (1) including validation with/without address and reverse authorization as part of the authorization procedure, and (2) splitting Authorization into two processes – validation and authorization itself. By comparing scores from Table 3 and Table 4, we illustrate the importance of standardizing the description of required functionality, and providing a more formal definition of scenarios.

Now, analyzing architectural compatibility, Figure 12 shows the COTS candidate (IBiz) as a supplier of services required by the E-payment system. Similarly to our requirements decomposition (Figure 11), the COTS candidate has been AND-decomposed into its main functionalities – interface, protocol and data management. In this example, we only analyzed protocol management because interface management concerns to other stages of the procedure, and data management is not required to the COTS component we are looking for. Therefore, features analyzed on COTS candidates are limited to how COTS components implement the authorization and capture by themselves.

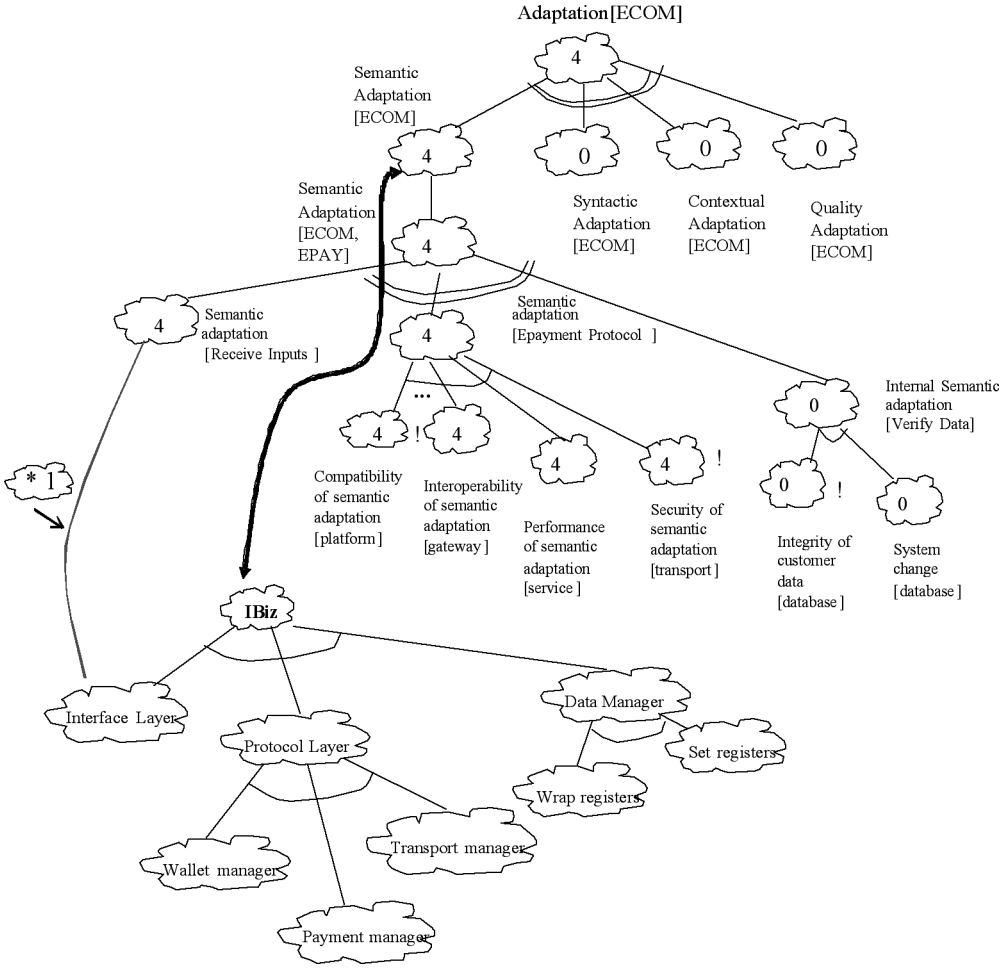

Figure 12: Architectural adaptability measures for a suitable COTS candidate (IBiz)

Then, impact on stability should be determined. To do so, it can be seen that Semantic Adaptation[ECOM, EPAY] has a metric of 4, which is the maximum score that Semantic Adaptation can get (from Figure 11). Claims from Figure 12 (noted as *1) indicate that the metric of 4 given to the interface service means fully pre-filtering compatibility. Therefore, from the scores of Semantic Adaptation[ECOM,EPAY], we may assume that the impact on stability is significant and this component might be filtered as a candidate to be further evaluated.

4.4 Improve and Control

Different interpretations of what is relevant for a particular case, and what is considered as a supporting function introduce ambiguity to the calculation process. As a consequence, our measures are affected by a particular scenario's description since calculation refers to the number of functions – without further discussion about their particular specification. For example, in our case study, "validation with address" and "reverse authorization" could be considered as part of an ordinary credit card authorization process, and hence considered as functional units.

As another example, we could choose a more detailed description of the functionality and decompose Authorization into "Credit Card Validation" and "Credit Card Authorization". In this case, calculation of metrics on provided and missed functionality would be different, and contribution would show which components partially contribute to reach credit card authorization.

The Analyze Phase would help us to improve the selection and probably the filtering process itself. For example, from Tables 3 and 4, note that components providing all required functionality remain unchanged on both tables; and only four components provide authorization and capture as required in our case (4/12 = 33%). It would indicate that searching a catalogue by category is not enough to find appropriated components. In our example, better categorizations would help distinguish credit card validation from authorization. Also it is interesting to note that from the 12 components, only one of them (IBiz) resulted architecturally compatible with our Java-based host architecture. Of course, the existence of only one suitable component might suggest that a decision should be made on searching for more compatible candidates (probably from other web portals and references). In any case, the suitable component should be similarly analyzed taking into account all features detailed on the adaptability decomposition.

5. LESSONS LEARNED AND MANAGERIAL IMPLICATIONS

Processes or methods to select COTS components may vary depending on different contexts – experience of the team with the candidate components, experience with a COTS-based development, and so forth. Particularly, from our case study some lessons learned have emerged as follows.

1. *Early measurement of functional suitability can reduce the number of candidates allowing a more objective value for decision-making*. Our case study shows that measurement is possible at early stages by analysing information of COTS components: from twelve COTS candidates, only four were pre-selected to be subjected to further analysis. Many conflicts resulted from the fact that the supplier and the composer focused the COTS component information differently. However, on the basis of the case analysis, the supply consisted of offerings of different suppliers, which could be compared in terms of functionality of the component. This means that although a low level of standardization exists with regard to the component's description, an example of matching was possible.

2. *Pre-filtering would result in a more effective resource allocation*. Although analysing selection effort was out of the scope of our study, we broadly analysed the impact of our process on a

filtering activity. Our case study was run during one week, considering that the average time to find mappings and calculate metrics for each COTS candidate was between 3-4 hours. However, the most interesting part was the iterative nature (Six Sigma-Based) of the process, which allowed us to dynamically analyse supplied functionality as well as requirements from the system. Requirements could be changed and improved based on a quick review (pre-filtering), saving time and effort that would have been invested on a deeper analysis of unsuitable candidates. We should note here that our conclusion strongly depends on the composer's experience and background as well as the quality of the requirement specification. Of course, results would not be the same with low-qualified composers, or confusing requirement specifications.

3. *Early detection of functionality requires that standards on how COTS components are documented be reinforced*. Information gathered from COTS candidates ranged from description of methods and properties in natural language to description by programming code. Certainly, this fact introduces ambiguity in some cases, complicates reading in some others, increases understanding effort, and makes that actual effectiveness of the filtering process be dependent on the composer's expertise to detect candidates. When establishing a component marketplace, one of the specific demands is to *provide well-structured information about components*, i.e. a well-structured catalogue. Then, a process to get vendor and product information should be carried out, where an immediate relationship is established between information provided by third parties and quality of the catalogue. This issue leads us to questioning about how information should be structured. For a detailed survey on recent proposals, see Cechich, Réquile-Romanczuk, Luzuriaga and Aguirre (2006).

4. *Composers' skills actually lead the search*. Of course, processes, techniques, and supporting tools are being defined to improve the filtering process. However, they still rely on how a composer perceives requirements and offerings. Although every human-intensive process – such as the ones involved in software engineering – is always affected by human perception, ambiguity of the processes is usually decreased by using particular notations and standards. Unfortunately, they are not available for COTS component searching and filtering yet. Deliverables from our process are a contribution in this sense. They helped us to reduce ambiguity of information to be analysed and compared from several candidates at early stages. They also provide us a framework for decision-making, where all candidates were evaluated by using similar measures. Finally, the defined deliverables allowed us to focus on possible improvements under the Six Sigma approach (i.e. improving scenarios' description).

5. *Assessing vendor's reliability is as much important as the identification of functional candidates itself*. We only have to look at our case study and the componentSource[4] catalogue again. From there and browsing the Web, we realised that the Bahs Software company is not a component supplier any more, and consequently AcceptOnline, CCProcessing, and CCValidate were withdrawn from the marketplace. Additionally, the Energy Programming Ltd. has changed its EnergyCreditCard component by the "StarFish EFT" component, which is an ActiveX component for validating credit card data. And PaymentCardAssist by Aldebaran has evolved into a complete suite called "Internet Commerce Toolkit". Then, from eleven COTS candidates in October 2004 (we exclude here the ComponentOneStudio), only five of them remain

[4] www.componentsource.org

available, and only one is suitable for our case – note that the IPWorks component is now only supplied as IBiz component. Then, our case study seems typical enough to sustain this lesson. We are aware that further research is needed to identify criteria for filtering. The point here is to find the right set of elements to make filtering effective but at the same time fast. In our proposal, we started from functional suitability and architectural adaptability as criteria for filtering; however possible extensions might include vendor's reliability and component's cost as well.

6. *Classification is not straightforward*. As we have seen, our twelve components were catalogued as members of the "Credit Card Authorization" group by the componentSource organization. From them, three components supply authorization and capture; seven components supply credit card validation; and one component – ComponentOneStudio – supplies any kind of functionality, not necessarily related to a credit card payment. Therefore, better classifications are needed. They might help facilitate the filtering process by using information about the catalogue itself. However, producing well-structured catalogues is not an easy task – component's granularity and several possibilities in classification make the process difficult.

From our lessons learned, managerial implications come from two points of views. On one hand, the COTS market itself: our research provided us a deeper understanding of the current situation of the COTS software component market. It is not merely about creating functional marketplaces or technical standards for the industry, it is more about composers and suppliers being interested in operating in relation to a market process and willing to adapt their offerings or needs. In other words, a COTS component filtering (and selection) is not only about the scope and definition of the component, but also about buyers and suppliers sharing the same trading rules. On the other hand, managing the filtering process brings internal implications to composers' as well as suppliers' organisations.

From the composer perspective, implications are on the need of establishing procedures to define and balance searching goals. Since negotiation of goals is part of a filtering process inherently, incorporating early measurement might facilitate discussions; i.e. rationale will be more explicit and funded. However, routinely negotiating and measuring imply that organisations have understood the COTS component filtering as a commercial trading problem, in addition to a technical one. Vendor viability and maturity are part of this managerial view. Although this implication is not new for COTS component trading, our research reinforces their importance.

From the supplier perspective, adequacy to requirements of a market should guide offerings but might promote other features as well. As a consequence, suppliers should be able to distinguish contributions, prioritise them, and sustain their permanence in the market. Measurement at early stages implies that suppliers would be able to quantify contribution; and hence recognize business opportunities more easily. Of course, our whole managerial view implies that we should move into a COTS component market, in which improvement is the key element to sustain successful businesses.

While the fundamental objective of the Six Sigma approach is the implementation of a measurement-based strategy that focuses on process improvement, some work is still needed to manage specific software projects. This is the case of COTS-based software development. Precisely here, our approach becomes a contribution by introducing quality from the beginning. Our tools aim at supporting a COTS-based development during its early stages allowing us to discover data that match software requirements and component services. However, matching provided and required

services requires not only standardizing information from vendors but also standardizing requirements for searching. Moreover, some proposals indicate that the necessity of formal processes for evaluation depends on the context, but the results also confirm the necessity of accelerating the identification and filtering of candidates through the use of knowledge-based portals (Mielnik, Lang, Lauriére, Schlosser, and Bouthors, 2003). Complexity of filtering will be undoubtedly affected; however, the process still might be guided by Six Sigma precepts allowing us to think of quality from the beginning.

6. CONCLUSIONS AND FUTURE WORK

The adoption of COTS-based development brings with it many challenges about the identification and finding of candidate components for reuse. To improve the process, we took into account how to identify suitable COTS components providing an early measure for comparison. We also considered that the evaluation of COTS candidates demands some inexact matching. Then, the Six Sigma-based phases of our proposal were further defined by introducing some techniques and measures, which would help in establishing a basis for reducing risks.

Filtering COTS components needs to ensure, as in any Six Sigma project, that decisions are based on facts and that customer's requirements have been considered. However, in the continuing attempt to introduce CBSD, organisations have problems identifying the content, location, and use of diverse components. Component composition requires access to tangible and intangible assets. Tangible assets, which correspond to documented, explicit information about the component, can vary from different vendors although usually include services, target platform, information about vendors, and knowledge for adaptation. Intangible assets, which correspond to tacit and undocumented explicit information, consist of skills, experience, and knowledge of an organisation's people. Six Sigma might help to put all these pieces together and define a measure-based procedure for pre-evaluating COTS components. But collecting effective measures is highly dependent on the amount and quality of information provided by third parties. Closing the gap between the required and provided information also implies dealing with standard information for analysis. On these lines, we are currently extending the filtering process to identify some key elements towards a knowledge-based process for COTS component identification (Réquilé-Romanczuk, Cechich, Dourgnon-Hanoune, Mielnik, 2005).

7. ACKNOWLEDGEMENTS

We thank anonymous reviewers for their valuable comments. This work is partially supported by the CyTED (Ciencia y Tecnología para el Desarrollo) project (506AC0287-COMPETISOFT), the UNComa project 04/E059, and the ESFINGE project supported by Spain (Dirección General de Investigación del Ministerio de Educación y Ciencia, TIN2006-15175-C05-05)

REFERENCES

ABTS, C. (2002): COTS-based systems (CBS) functional density: a heuristic for better CBS design. In *Proceedings of the 1st International Conference on COTS-Based Software Systems*, volume 2255 of LNCS. Orlando, Florida, Springer-Verlag: 1–9.

ALBERT, C. and BROWNSWORD, L. (2002): Evolutionary process for integrating COTS-based systems (EPIC): An Overview. *Technical Report 20030TR-009*, SEI, CMU.

ALEXANDER, R. and BLACKBURN, M. (1999): Component assessment using specification-based analysis and testing. *Technical Report SPC-98095-CMC*, Software Productivity Consortium.

ALLEN, P. and FROST, S. (2001): Planning team roles for CBD. In component-based software engineering – Putting the pieces together, Addison-Wesley.

AYALA, C., BOTELLA, P., and FRANCH, X. (2005): On goal-oriented COTS taxonomies construction. In *Proceedings of the Fourth International Conference on COTS-Based Software Systems*, volume 3412 of LNCS, Springer Verlag: 90–100.

BERTOA, B. TROYA, J. and VALLECILLO, A. (2003): A survey on the quality information provided by software component vendors. In *Proceedings of the ECOOP QAOOSE Workshop*.

BIANCHI, A., CAIVANO, D., CONRADI, R., JACCHERI, L., TORCHIANO, M. and VISAGGIO, G. (2003): COTS products characterization: Proposal and empirical assessment. In *Proceedings of ESERNET 2001-2003*, volume 2765 of LNCS, Springer Verlag: 233–255.

BIEHL, R. (2004): Six sigma for software. *IEEE Software*, 8(2): 68–70.

BOEHM, B. (1998): A spiral model of software development and enhancement. *IEEE Computer*: 61–72.

BOEHM, B., BOSE, P., HOROWITS, E. and LEE, M. (1998): Software requirements negotiation aids: A theory-W based spiral approach. In *Proceedings of the 17th International Conference on Software Engineering*: 243–253.

BRAGA, R., MATTOSO, M. and WERNER, C. (2001): The use of mediation and ontology technologies for software component information retrieval. In *Proceedings of the 2001 symposium on Software reusability: putting software reuse in context*, ACM press, Ontario, Canada: 19–28.

CECHICH, A. and PIATTINI, M.(2003): Defining stability for component integration assessment. In *Proceedings of the Fifth International Conference on Enterprise Information Systems*, Angers, France: 251–256.

CECHICH, A. and PIATTINI, M. (2004a): Managing COTS components using a six sigma-based Process. In *Proceedings of the Fifth International Conference on Product Focused Software Improvement*, volume 3009 of LNCS, Nara, Japan, Springer-Verlag: 556–567.

CECHICH, A. and PIATTINI, M. (2004b): Balancing stakeholder's preferences on measuring COTS component functional suitability. In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, Porto, Portugal: 115–122.

CECHICH, A. and PIATTINI, M. (2004c): On the measurement of COTS functional suitability. In *Proceedings of the Third International Conference on COTS-Based Software Systems*, volume 2959 of LNCS, Los Angeles, USA, Springer-Verlag: 31–40.

CECHICH, A. and PIATTINI, M. (2004d): Quantifying COTS component functional adaptation. In *Proceedings of the Eight International Conference on Software Reuse*, volume 3107 of LNCS, Madrid, Spain, Springer-Verlag: 195–204.

CECHICH, A. and PIATTINI, M. Piattini (2005a): Filtering COTS components through an improvement-based process. In *Proceedings of the Fourth International Conference on COTS-Based Software Systems*, volume 3412 of LNCS, Bilbao, Spain, Springer-Verlag: 112–121.

CECHICH, A. and PIATTINI, M. (2005b): Measurement of COTS functional suitability: an E-payment case study, In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, Miami, USA.

CECHICH, A. and PIATTINI, M. (2006): Early detection of COTS component functional suitability. *Information and Software Technology*, Elsevier (article in press).

CECHICH, A., PIATTINI, M. and VALLECILLO, A. (Eds) (2003): Component-based software quality: Methods and techniques, volume 2693 of LNCS, Springer-Verlag.

CECHICH, A., RÉQUILE-ROMANCZUK, A., LUZURIAGA, J. and AGUIRRE, J. (2006): Trends on COTS component identification. In *Proceedings of the Fifth International Conference on COTS-Based Software Systems*, IEEE Computer Science Press.

CHUNG, L., NIXON, B., YU, E. and MYLOPOULOS, J. (2000): Non-functional requirements in software engineering. Kluwer Academic Publisher, 2000.

CHUNG, L. and SUBRAMANIAN, N. (2001): Process-oriented metrics for software architecture adaptability. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*: 310–312.

COCKBURN, A. (2001): Writing effective use cases. Addison-Wesley.

DAVIS, L., GAMBLE, R. and PAYTON, J. (2002): The impact of component architectures on interoperability. *Journal of Systems and Software*, (61):31–45.

DE FEO, J. and BAR-EL, Z. (2002): Creating strategic change more efficiently with a new design for six sigma process. *Journal of Change Management*, 3(1):60–80.

GACK, A. and ROBINSON, K. (2003): Integrating improvement initiatives: Connecting six sigma for software, CMMI, personal software process and team software process. *Software Quality Journal*, 5(4):5–13.

HOLMES, L. (2004): Evaluating COTS using function fit analysis, Q/P management group INC., http://www.qpmg.com

JACCHERI, L. and TORCHIANO, M. (2002): A software process model to support learning of COTS products. *IDI NTNU Technical Report*.

JACOBSON, I., BOOCH, G. and RUMBAUGH, J.(1999): The unified development process, Addison-Wesley.

KAIYA, H., HORAI, H. and SAEKI, M. (2002): AGORA: Attributed goal-oriented requirements analysis method. In *Proceedings of the IEEE Joint International Conference on Requirement Engineering*: 13–22.

KAIYA, H. and SAEKI, M. (2004): Weaving multiple viewpoints specifications in goal oriented requirements analysis. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, IEEE Society Press: 418–427.

LIPSON, H., MEAD, N. and MOORE, A. (2002): Can we ever build survivable systems from COTS components? In *Proceedings of CaiSE 2002*, volume 2348 of LNCS, Springer-Verlag: 216–229.

MAIDEN, N. and NCUBE, C. (1998): Acquiring COTS software selection requirements, *IEEE Software* 15(2):46–56.

MARCA, D. and McGOWAN, C. (1988): SADT: Structured analysis and design technique, McGraw-Hill Co.

MIELNIK, J-C., LANG, B., LAURIERE, S., SCHLOSSER, J-G. and BOUTHORS, V. (2003): eCots platform: An inter-industrial initiative for COTS-related information sharing. In *Proceedings of the Second International Conference on COTS-Based Software Systems*, volume 2580 of LNCS, Ottawa, Canada, Springer-Verlag: pages 157–167.

NCUBE, C. and MAIDEN, N. (1999): Guiding parallel requirements acquisition and COTS software selection. In *Proceedings of the IEEE International Symposium on Requirements Engineering*: 133–141.

PAHL, C. (2003): An ontology for software component matching. In *Proceedings of the Sixth International Conference on Fundamental Approaches to Software Engineering*, volume 2621 of LNCS, Warsaw, Poland, Springer-Verlag: 6–21.

RÉQUILE-ROMANCZUK, A., CECHICH, A., DOURGNON-HANOUNE, A. and MIELNIK, J-C. (2005): Towards a knowledge-based framework for COTS component identification, In *Proceedings of the 2nd ICSE International Workshop on Models and Processes for the Evaluation of OTS Components* (MPEC), Saint Louis, USA.

ROLLAND, C., SOUVEYET, C. and BEN ACHOUR, C. (1998): Guiding goal modelling using scenarios. *IEEE Transactions on Software Engineering*, 24(12):1055–1071.

TAYNTOR, C. (2002). Six Sigma Software Development. Auerbach Publications.

TYSON B., ALBERT, C. and BROWNSWORD, L. (2003): Improving processes for commercial off-the-shelf-based systems. *The Journal of Defense Software Engineering*: 17–22.

TORCHIANO, M. and MORISIO, M. (2004): Overlooked aspects of COTS-based development. *IEEE Software* 21(2):88–93.

YIN, R. (1994): *Case Study Research: Design and Methods*. In Applied Social Research Methods Series. Sage Publications Inc. Thousand Oaks, CA, 2nd Edition.

BIOGRAPHICAL NOTES

Alejandra Cechich

Alejandra Cechich received a European PhD in Computer Science from the University of Castilla-La Mancha, Spain and the MSc in Computer Science from the University of South, Argentina. She is an associate professor at the University of Comahue, Argentina, where she leads the GIISCo Research Group. Her interests are centered on conceptual modeling, software quality, and component technology and their use in the systematic development of software systems. She is a member of ACM and IEEE Computer Society. Contact details: Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Buenos Aires 1400, 8300, Neuquén, Argentina; acechich@uncoma.edu.ar.

Mario Piattini

Mario Piattini is a full professor at the UCLM. His research interests include software quality, metrics and maintenance. He gained his PhD in Computer Science at the Polytechnic University of Madrid, and he leads the Alarcos Research Group. He is CISA and CISM by ISACA. He is a member of ACM and the IEEE Computer Society. Contact details: Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Mario.Piattini@uclm.es.