SOAs factors, criteria and metrics. SERP 2012

R. Belkhatir¹, M. Oussalah¹, and A. Viguier²
¹Department of Computer Science, University of Nantes, Nantes, Loire Atlantique, France
²Research and Development Department, BeOtic, Rezé, Loire Atlantique, France

Abstract – The current paper presents a semi-automated method for evaluating SOAs called SOAQE. This method corrects defects observed so far such as lacks of pertinence and accuracy for evaluation results. SOAQE takes as a starting point the McCall model, describing software quality, which led to an international standard for the evaluation of software quality (ISO 9126). This model is organized around three types of quality attributes (factors, criteria and metrics). The method consists in decomposing the whole architecture and evaluating it according to the McCall model, i.e. a list of quality factors arising from business needs, grouping criteria composed by metrics. Our experimentations led us to quantify numerically a first determining factor for SOAs, the ‘dynamism’ and some attributes of its structure: namely the ‘loose coupling’ criterion and its constituent metrics (‘physical, semantic and syntactic’).

Keywords: SOA, quality, evaluation, factor, criteria, metric.

1 Introduction

An architectural paradigm defines groups of systems in terms of models of structures; component and connector vocabularies and rules or constraints on relations between systems [1]. We can distinguish a few architectural paradigms for distributed systems, and, among the most noteworthy ones, three have contributed to the evolution of the concerns. These are chronologically object oriented architectures (OOA), component based architectures (CBA) and Service oriented architectures (SOA). First developers were quickly aware of code repetitions in applications and sought to define mechanisms limiting these repetitions. OOA provides great control of the reusability (reusing a system the same way or through a certain number of modifications) which paved the way to applications more and more complex and consequently to the identification of new limits in terms of granularity. These limits have led to the shift of the concerns towards the composability (combining in a sure way its architectural elements in order to build new systems or composite architectural elements). Correlatively, the software engineering community developed and introduced CBA to overcome this new challenge and thus, the CBA reinforces control of the composability and clearly formalizes the associated processes. By extension, this formalization establishes the base necessary to automation possibilities. At the same time, a part of the software community took the research in a new direction: the dynamism (developing applications able to adapt in a dynamic, automatic and autonomous ways their behaviors to answer the changing needs of requirements and contexts as well as possibilities of errors) concern as the predominant aspect. SOA has been developed on the basis of the experience gained by objects and components, with a focalization from the outset on ways of improving the dynamism. Developing an SOA involves many risks, so much the complexity of this technology is notable (particularly for services orchestration) [2]. First and foremost among these, is the risk of not being able to answer favorably to expectations in terms of quality of services because quality attributes directly derive from business objectives. As these risks are distributed through all the services, the question of evaluating SOA has recently arisen. This is to identify and correct remaining errors that might have occurred after the software design stage and, implicitly, to reduce subsequent risks. Lots of tools have been created to evaluate SOAs but none of them clearly demonstrated its effectiveness [3]. The SOAQE method presented in this paper allows evaluating SOAs by combining the computerized approach and the human intervention to eliminate lacks identified with past methods. The main idea resides in automating the process to avoid time-wasting evaluations. The process consists in three principal stages detailed later in this paper. In the current paper, we first relate in the section 2 the interests of the evaluation and we also analyze the existing works. We present the SOAQE method and the stakeholders participating to the evaluation in the section 3. We finally relate the experimentation which has been done by the lab team in the section 4. We conclude this paper with a discussion comparing the SOAQE method to past ones.

2 State of the art

The SOA evaluation relates to qualitative and quantitative approaches, load prediction associated with evolutions and theoretical limits of a given architecture.

2.1 Related works on SOA evaluation

From this perspective, tools and existing approaches have shown their limitations for SOA [3]. We are currently attending the development of a new generation of tools developed by industrialists in a hand-operated way [4]. The
scale of the task has brought the academic world to tackle these issues and to try to develop a more formal and generic approach than different existing methods (ATAM, SAAM) to evaluate SOAs [3].

In any software architecture evaluation, three activities are critical to success:
1. Specifying the quality attributes deriving from the commercial needs.
   ➢ The current paper focuses on a finite set of technical attributes having significant impacts on the global quality, from the development process to the system produced as outcome [5].
2. Selecting a representative assembly of stakeholders for the evaluation.
   ➢ In this paper, we work with a restricted and technical assembly of stakeholders listed in section three.
3. Describing the architecture in a clear, expressive and understanding way.
   ➢ The present paper does not treat this part of the problematic because it has always been covered by the lab team in a previous paper [6].

### 2.2 Evaluation Results

In concrete terms, SOA evaluations produce a report which form and content vary depending on the method used. But, in general terms, the evaluation generates textual information and answers two types of questions [3]:

- **1.** Is the architecture adapted to the system for which it has been conceived?
- **2.** Is there any other architecture more adapted to the system in question?

---

- **1.** It could be said that the architecture is adapted if it favorably responds to the three following points:
  - **i.** The system is predictable and could answer to the quality requirements and to the security constraints of the specification.
  - **ii.** Not all the quality properties of the system result directly from the architecture but a lot do; and for those that do, the architecture is deemed suitable if it makes it possible to instantiate the model taking into account these properties.
  - **iii.** The system could be established using the current resources: the staff, the budget, and the given time before the delivery. In other terms, the architecture is buildable.

This definition will open the way for all future systems and has obviously major consequences. If the sponsor of a system is not able to tell us which are the attributes to manage first, well, any architecture will give us the answer [3].

- **2.** A part of SOA evaluation consists in capturing the quality attributes the architecture must handle and to prioritize the control of these attributes. If the list of the quality attributes is suitable in the sense that at least all the business objectives are indirectly considered, then, we can keep working with the same architecture. Otherwise, it is time work with architecture more suitable for the system. These quality attributes may be conflictive for achieving business objectives. In such a case, the project manager must take the decision to focus on a limited set of attributes, especially if the evaluation of the architecture gives a positive result in a sector and a poor one in another sector [7]. However, provided that it is still able to favorably answer to non-functional and functional requirements of the specification, the architecture is able to exist with these rare failures. Moreover, since each sector is linked to a list of objectives and, these latter are pursued by focusing on certain quality attributes; the best way to reinforce the neglected sectors is to produce a more robust work on attributes correlated to the sector in question. The evaluation will help to indicate where the architecture fails, but the balance between the cost of the evaluation and the help it provides to ameliorate the project remains relative to any architecture. Then, an architecture evaluation does not provide answers to whether or not the architecture is adapted; if it is "good" or "bad" or if it is rated "seven out of ten"; it only tells us where the danger is. The evaluation could be applied to a single SOA or to a group of several SOAs competing to be chosen for the final system. In this last case, it first identifies the relevant business objectives needed in the comparison and then, examines available documentation for each architecture candidate. Then, it finally scores the fitness of the candidate architecture, summarizes the analysis results and provides a recommendation for the decision-making process.

### 2.3 Measuring the quality

It has been suggested that software production is out of control because we cannot quantitatively measure it. As a matter of fact, Tom DeMarco (1986) stated that "you cannot control what you cannot measure" [8]. The measurement activity must have clear objectives and a whole set of sectors need to be measured separately to ensure the right management of the software. For example, we know that the administrator must measure the software quality in order to compare projects, make previsions and set reasonable improvement objectives. In order to bring these operations to fruition, the scientific community utilizes models of quality introducing what we call software attributes decomposition.

#### 2.3.1 McCall model

One of the models that have been published is the McCall model in 1977 decomposing quality attributes in three stages. This model led to the IEEE standard: ISO/IEC 9126. A certain number of attributes, called external (applicable to running software), are considered as key attributes for quality. We call them quality factors [3] (for example, the reliability: ability of a system to keep operating over time).
attributes are decomposed in lower level attributes, the internal attributes (do not rely on software execution), called quality criteria (for example, the testability is one of the "maintainability" criteria). And each criterion is associated to a set of attributes directly measurable and which are called quality metrics (for example, the testability criterion is measured by the statement coverage: evaluation of the achievable instructions percentage exerted, the connections coverage: evaluation of the active connections percentage, etc.).

2.4 Lack of precision

Current methods of evaluation stop the quality attributes decomposition at the "quality factors" step and thus remain too vague when it comes to giving accurate measures. These methods are not precise because they cannot go further in the decomposition and consequently they cannot be automated to the point of defining a finite value for each attribute. They lead inevitably to the establishment of a brainstorming between stakeholders (see section 3.2.4) for the purpose of the institution of a utility tree. Because stakeholders do not have tools to quantitatively measure each quality factor chosen, they shall set up scenarios aiming to respectively solicit separately each criterion, and factor. An approximated evaluation of the architecture is then realized after having studied the system behavior while carrying out the scenarios set.

3 SOAQE

It is precisely where our work differs from those existing insofar as we wish to obtain a precise quantitative measurement for each quality factor with the SOAQE method. We especially aim to automate the process in order to avoid hand-operated evaluations pushing to solicit stakeholders for the whole evaluation.

3.1 Principle of the model

The process consists in three principal stages.

Each corresponds to a decomposition step of our quality attributes. We first identify decisive quality factors for our architecture. Then we isolate the quality criteria defining them. And finally we define quality metrics composing each criterion in order to quantify them numerically.

3.2 Steps in more details

The main idea of the SOAQE process is to evaluate in three steps the whole architecture from every metric to the set of quality factors obtained after having previously identified the business objectives. Our work is based on the architect point of view and the attributes selected are the ones considered as the most relevant among all existing.

3.2.1 Quality factors

Some authors have defined the CBA with reusability and composability [9]. Basing on previous analysis, we define the SOA with the Reusability, the Composability and the Dynamism.

These three attributes, that we identified as quality factors for SOA represent the qualitative quintessence which has directed the definition of the object, component and service paradigms. The figure 2 illustrates this analysis and offers a high-level vision of the SOA interest points.

![Figure 2: SOA interest points](image_url)

The first step of the SOAQE method consists in choosing a first quality factor to study in depth and there exist a lot which could come out after the analysis of the business objectives. But we have naturally chosen to work on those identified as the qualitative quintessence for SOA. These three quality attributes (dynamism, composability and reusability) define each of our three paradigms to varying degrees. Moreover, there exist a hierarchical ranking propelling “dynamism” on top of SOA concerns, and this is precisely
why, we chose to especially focus deeply on this quality factor. We may record that each of the two others attributes is of major importance for the three paradigm considered (object, component or service oriented architectures).

3.2.2 Quality criteria and quality point of views (QPV)

With regards to our work and after having identified the determining factor quality for SOA (i.e. the dynamism), we were interested in the second step of the SOAQE process, namely discovering the criteria defining the factor on which we have gone through.

Further down the road, any factor is composed by a lot of criteria that could be looked at as part of our work.

There exist quality points of views (QPV) which group together criteria according to their characteristics. The QPV notion is fundamental because it allows company to target their evaluation on selected families of criteria and these families of criteria are common to all existing architectures.

We deliberately concentrated our work on a technical QPV (see figure 3) grouping technical criteria because we adopted the point of view of an architect (a technical stakeholder). In this light, we identified six criteria common to each of our three factors. These technical criteria gather elements having significant impacts on global quality, from the development process to the system produced.

**Loose coupling**: Potential of dependences reduction between services and dependences between processes.

**Explicit architecture**: Paradigm ability to define clear architectural application views, i.e. providing the means of identifying and clarifying functionalities associated to services composing the system.

**Expressive power**: Potential of paradigm expression in terms of creation capacity and optionalities. It is based on the number of processes provided to specify, develop, handle, carry out and implement services.

**Communication abstractions**: Paradigm capacity to abstract services functions communications.

**Upgradability**: Paradigm ability to make evolve its services (based on processes supporting these evolutions).

**Owner's responsibility**: Corresponds to the responsibilities sharing out between services providers and consumers. These responsibilities are focused on the software entity re-used in terms of development, service quality, maintenance, deployment, execution and management. This distribution expresses the degree of freedom granted to service consumers by the service provider.

Each of these quality criteria is given varying degrees of consideration according to the quality factor in question. Our previous works [5] allowed organizing hierarchically (under three distinct levels) these quality criteria for each of the three quality factors (dynamism, reusability and composability).

Consequently, we obtain the triptych of the figure 4 while considering all paradigms.

**Figure 4**: Expression of reusability, composability and dynamism perspectives.

While focusing on the dynamism, identified earlier as being the key quality factor for SOA, our previous work [5] allowed to conclude that the “loose coupling” criteria is of biggest importance for this factor (see figure 4), this is why we chose to concentrate in depth, on it whereas, the criterion “expressive power” is of less importance.
3.2.3 Quality metrics

The coupling is relatively well known by the community, and thus, our lab team members decided to look into the matter [5]. Correlatively, we found three quality metrics for the latter which must be considered for the third and final step of the SOAQE process, that is to say, the quantification and the evaluation of all the metrics (The semantic coupling: based on the high-level description of a service defined by the architect, the syntactic coupling: measures dependencies in terms of realization between abstract services and concrete services and the physical coupling: measures dependencies between concrete services really utilized, in collaboration and in execution). In this light, we can draw a triptych clearly presenting the metrics extracted and levels of acceptance for each of them (see figure 5).

Semantic coupling

High coupling: The service takes part in an essential functionality of the composite.
Low coupling: The service takes part in a nonessential functionality of the composite. The global quality is not guaranteed anymore if one or more from these functions are withdrawn. We pose that if all these nonessential functions disappear the composite becomes unusable.
Non-predominant coupling: An abstract service and a composite are in non-predominant coupling if this service takes part in a nonessential function of the composite and if the withdrawal of this function does not have any significant impact on the global quality.

Syntactical coupling

High coupling: An abstract service is in high syntactical coupling with its concrete solution if this solution (a concrete service or a composition of concrete services) represents the single possibility of realization.
Weak coupling: An abstract service and a concrete service are in weak coupling if there are several concrete alternatives to the realization of this abstract service.

Physical coupling

The physical coupling focuses on the implementation of the service. This implementation corresponds to a particular instance of the service where a choice has been made concerning concrete services to use. A unique solution has been chosen to fulfill each of the needs expressed by abstract services. It reuses existing researches [10] and it is based on measurements such as methods calls, messages exchanged, the number of linked services, commune objects and so forth. These metrics shall make it possible to identify physical dependencies between concrete services.

3.2.4 Stakeholders

Further down the road, each of the SOAQE method steps require the intervention of external evaluators (called stakeholders) to set the coefficients prioritizing factors, criteria and metrics considered. The categories of stakeholders are often the same for the SOA systems, especially when we work with criteria from the same family. Here is a technical list of stakeholders chosen for evaluating technical attributes.

![Figure 5: Loose coupling metrics](image)

We have the software architects which main responsibilities include experimenting with and deciding between different architectural approaches, the developers which main responsibilities include implementing the architectural elements of the system according to the architecture specification; the integrators which main responsibilities are to ensure that the architecture and implementation conform to open and widely accepted standards; the maintenance developers which main responsibilities include modifying the software to correct defects and adapting the software when environmental changes occur (e.g., hardware or operating system changes). We also have the developers of service users: these external developers may provide input on application program interface (API) design and desired quality of service and finally the external developers of service providers: they may contribute requirements for interaction with their services, as well as knowledge of qualities and limitations of their systems.

3.2.5 Coefficients

Concerning the first step of SOAQE, coefficients assigned to the factors will depend on the company needs. Our works led us to conclude that for SOA and the three factors we worked with, we would allocate, according to our hierarchical ranking, a coefficient of ‘3’ for the “dynamism” whereas we would affect the value ‘2’ for the “reusability” and the “composability”.

With regards to the second step, our works led to list the six technical quality criteria chosen under three distinct levels of acceptance, α, β and γ at which we assign respectively the values ‘3, 2 and 1’, consequently, the “loose coupling”, the “upgradability” and the “communication abstraction” will be
allocated the value ‘3’. The coefficient ‘2’ goes for the “owner’s responsibility” and “explicit architecture” criteria and ‘1’ for the “expressive power”.

And finally, the three metrics studied may be all assigned to the value ‘1’ meaning that they are equally important for calculating the global coupling of SOAs. These coefficients will be used as a basis for the following section (Experimentation). They have been affected to quality attributes as an example; however, these latter have been chosen according to the principle of proportionality validated by the lab-team. We can select other impact coefficients providing that we keep the same proportionality between the quality-attributes considered.

4 Experimentation

For the experimentation, we tempted to quantitatively measure the key quality attributes discussed in the previous sections of this paper; notably, the quality factor “dynamism”, the “loose coupling” criteria and the “physical, syntactic and semantic coupling” metrics. That being said, it is important to note that the SOAQE method must be produced for every quality factor identified after having analyzed the objectives of the company and the set of criteria and metrics belonging to that quality factor.

4.1 Loose Coupling

Taking as a starting point an existing formula of the field of “Preliminary analysis of risks” (see formula 6.1) [11], our works led to the identification of a mathematical formula (see formula 6.2) combining the three couplings studied: semantic, syntactic and physical.

NB: The simplified formula (see formula 6.1) usually used in the automotive industry, makes it possible to measure the default risk of a car component A is the Criticality of the car component, B is the Probability of occurrence of a failure on this component and C is the Probability of non-detection of this failure.

\[
R = A \ast B \ast C
\]  

\[
\text{Coupling} = \left\{ (a), (b), (c) \right\} \left\{ \left( \prod_{i=1}^{N} \sum_{j=1}^{I} \left( P_{ij} \right)^{\alpha} \right) \ast C_{phys} \right\} \ast P_{detect}
\]

We associate this concept of risk with our vision of the coupling. Correlatively, the quintessence of the coupling is the expression of the dependences which can exist between two elements and the principle of dependence defines that one element cannot be used without the other. Reducing the risk that the role defined by a service cannot be assured anymore is decreasing the dependence of the application in relation to this service and thus reducing its coupling. The calculation of this risk takes into account all the characteristics influencing the coupling by redefining the three variables A, B and C according to the semantic, syntactic and physical couplings. The global coupling corresponds to the sum of the three couplings calculated individually beforehand. The lower this result is, the more the coupling is weak.

NB: The criticality \( A \in \{(a),(b),(c)\} \) is affiliated to the semantic coupling. ‘a’ if the service is only associated to non predominant couplings, ‘b’ for non predominants and low couplings and ‘c’ for non predominants, low and high couplings, while ‘Ps’ is the probability of failure of a service.

This generic coupling formula can directly be used to quantify the quality of an architecture by weighting up each of the attributes concerned by means of the coefficients isolated after having hierarchised the attributes according to their importance. Indeed, as we already specified in section 2, we cannot automate this operation and define continuously the same coefficients for all the architectures considered because this operation is specific to the business objectives of the company.
composability, all three affected by their respective coefficients. The formula hereinafter allows calculating the whole quality of an SOA.

*NB: the lower the loose coupling result is, the more the coupling is weak. Conversely, the higher the architecture quality result is, the more the quality is good. The result of each criterion is expressed in percentage, this is why we subtract to 1 the result found.*

For any architecture considered, we are able to determine a finite value for the loose coupling criteria, the remaining work consists in defining a way to calculate the five others criteria in order to isolate a finite value for the quality.

## 5 Discussion

Because SOA implies the connectivity between several systems, commercial entities and technologies; some compromises regarding the architecture must be undertaken. Forasmuch as the decisions about SOA tend to be pervasive and, consequently, have a significant impact on the company; setting an evaluation of the architecture early in the life of the software is particularly crucial. During software architecture evaluations, we weigh the relevance of each problematic associated to the design after having evaluated the importance of each quality attribute requirement. The results obtained when evaluating software architectures with existing methods (ATAM, SAAM) are often very different and none of these latter carries out it accurately [12]. We know the causes of this problem: most methods of analysis and automatic quality evaluation of software systems are carried out from the source code; whereas, with regard to evaluation cases of architectural models, the analysis is conducted based on the code generated from the model. From this code, there exist calculated metrics, more or less complex, associated with algorithms, methods, objects or relations between objects. From an architectural point of view, these techniques can be indicated of low level, and can be found out of step with projects based on new complex architectures.

The evaluation concerns qualitative and quantitative aspects, the prediction of the load associated to evolutions and on the theoretical limits of a given architecture. These architectures evaluations can be made as well on architectures under development as on existing ones.

## 6 Conclusion

The finality of our work is to design a conceptual framework and, in fine, a semi-automated prototype (based on past methods, such as ATAM or SAAM) which could quantify with an accurate value the quality of the whole service oriented architecture. Another pursued goal consists in bringing to the customer "less abstract" documents than those proposed today. The quality concept remaining a relative one, we will target the sectors requiring a special attention by directly addressing the various development lab teams charged with the relevant functions.

## 7 References


