A model introducing SOAs quality attributes decomposition

Riad Belkhatir, Mourad Oussalah
Department of Computing
University of Nantes
Nantes, France
{riad.belkhatir, mourad.oussalah}@univ-nantes.fr

Arnaud Viguier
Department Research and Development
BeOtic
Rezé, France
arnaud.viguier@beotic.com

***Abstract*—Recently, service oriented architecture (*SOA*) has been popularized with the emergence of standards like Web services. Nevertheless, the shift to this architectural paradigm could potentially involve significant risks including projects abandonments. With this in mind, the question of evaluating SOA quality arose. The appearance of methods like ATAM or SAAM propelled software architecture evaluation to a standard stage for any paradigm. However, there still are a number of concerns that have been raised with these methods; in particular their cost in terms of time and money, essentially because of the hand-operated nature of the evaluations conducted. The model proposed in this paper for evaluating SOAs takes as a starting point the McCall model; it allows the whole architecture to be decomposed in three types of quality attributes (factor, criterion and metric).**

Keywords- SOA; factor; criterion; metric

I. INTRODUCTION

Architectural paradigms are design patterns for the structure and the interconnection between software systems [1]. Their evolution is generally linked to the evolution of the technology.

An architectural paradigm defines groups of systems in terms of:

- Model of structure.
- Component and connector vocabularies.
- Rules or constraints on relations between systems [2].

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 s**ervice oriented architectures (SOA).**

First developers were quickly aware of code repetitions in applications and sought to define mechanisms limiting these repetitions. **OOA** is focused on this concern and its development is one of the achievements of this research. 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** concern (*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*) as the predominant aspect. In short, **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.**

Service oriented architecture is a popular architectural paradigm aiming to model and to design distributed systems [3]. SOA solutions were created to satisfy commercial objectives. This refers to a successful integration of existing systems, the creation of innovating services for customers and cost cutting while remaining competitive. For the purpose of making a system robust, it is necessary that its architecture can meet the functional requirements ("*what a system is supposed to do*"; defining specific behaviors or functions) and the non-functional ones ("*what a system is supposed to be*"; in other words, the quality attributes) [4]. Furthermore, developing an SOA involves many risks, so much the complexity of this technology is notable (*particularly for services orchestration*). 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. Multi-million dollar projects, undertaken by major enterprises (Ford, GSA) failed and were abandoned. As these risks are distributed through all the services, the question of evaluating SOA has recently arisen. It is essential to carry out the evaluation of the architecture relatively early in the software lifecycle to save time and money [5]. 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

[6]. The model presented in this paper allows evaluating SOAs by combining the computerized approach and the human intervention. We first relate in the section 2 the state of the art and we present the model in the section 3. We relate the experimentation led by the lab team in the section 4 and we conclude the paper with a discussion comparing the past works and our model in the last section.

II. STATE OF THE ART

A. Related works on SOA Evaluation

There is something far more important with the SOA evaluation as it is the bond between business objectives and the system, insofar as evaluation makes it possible to assess quality attributes of services composing the system [4].

The evaluation relates to:

- Qualitative and quantitative approaches.
- Load prediction associated with evolutions.
- Theoretical limits of a given architecture.

From this perspective, tools and existing approaches have shown their limitations for SOA [6]. We are currently attending the development of a new generation of tools developed by industrialists in a hand-operated way. 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 [6]) to evaluate SOAs.

B. Evaluation Results

In concrete terms, SOA evaluations product 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 [6].

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:

a) The system is predictable and could answer to the quality requirements and to the security constraints of the specification

b) 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.

c) 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 quality attributes to manage first, well, any architecture will give us the answer [6].

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 (*each of which is related to specific business objectives*) 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 to restart from the beginning and to work with a new architecture, more suitable for the system.

C. 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" [7]. 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.

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 [6]. These attributes are decomposed in lower level attributes, the internal attributes (which do not rely on software execution), called quality criteria and each criterion is associated to a set of attributes directly measurable and which are called quality metrics.

D. From past works to our model.

Current methods of evaluation stop the quality attributes decomposition at the “quality factors step” and remain too vague when it comes to giving accurate measures to quality. 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. Our work differs from those existing insofar as we wish to obtain a precise quantitative measurement for each quality factor with our model.

III. THE MODEL PROPOSED

A. The model in more details.

The main idea of the 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. The process consists in three principal stages corresponding each to a decomposition step of our quality attributes.

1) We first identified relevant quality factors for our architecture:

a) the CBA is defined with reusability and composability [8]. *Basing on previous analysis, we define the SOA with the reusability, the composability and the dynamism. 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.

2) Then, we isolate the quality criteria defining them:

a) We concentrated our work on technical criteria because we adopted the point of view of an architect that is itself 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: the ***loose coupling*** *(potential of dependences reduction between services), the* ***explicit architecture*** *(paradigm ability to define clear architectural application views), the* ***expressive power*** *(potential of paradigm expression in terms of creation capacity and optionalities), the* ***communication abstractions*** *(paradigm capacity to abstract services functions communications), the* ***upgradability*** *(paradigm ability to make evolve its services), and the* ***owner's responsibility*** *(corresponds to the responsibilities sharing out between services providers and consumers).*

3) And finally we define quality metrics composing each criterion in order to quantify them numerically:

*a) Our previous work allowed to conclude that the "****loose coupling****" criterion is of biggest importance for the quality factor "dynamism"* [9]. *We found three quality metrics for the latter which must be considered for the last stage of our model (the semantic coupling: {high, low or non-predominant} based on the high-level description of a service defined by the architect, the syntactic coupling: {high, low} measures dependencies in terms of realization between abstract services and concrete services and the physical coupling: {γ, β and α with 0≤γ≤β≤α} focusing on the implementation of the service). These metrics shall make it possible to identify physical dependencies between concrete services.*

B. Coefficients

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 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'; We allocated the "loose coupling", the "upgradability" and the "communication abstraction" with 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. 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.

IV. 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 reproduced 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. Taking as a starting point an existing formula of the field of "Preliminary analysis of risks" (see formula 1.1) [10] our works led to the identification of a mathematical formula (see formula 1.2) combining the three couplings studied: semantic, syntactic and physical.

NB: The simplified formula (see formula 1.1) 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.

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∈[(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.

$$R = A * B * C \tag{1.1}$$

$$Coupling = \left\{\{(a),(b),(c)\}\right\} * \left\{\left(\left(\prod_{k=1}^{\lambda}\prod_{i=1}^{N}\sum_{j=1}^{i}((P_s)_{kij})^{\alpha_{kij}}\right) * C_{phys}\right)\right\} * \left\{P_{ndetect}\right\} \tag{1.2}$$

Figure 1: Defaut risk of a car component (1.1) and global coupling of an architecture (1.2) formulas.

This generic coupling formula can directly be used to quantify the quality of the architecture by weighting up each of the attributes concerned by means of the coefficients isolated after having organized 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. By applying to known quality attributes the coefficients determined in the section III.*B*, we obtain the following tree:

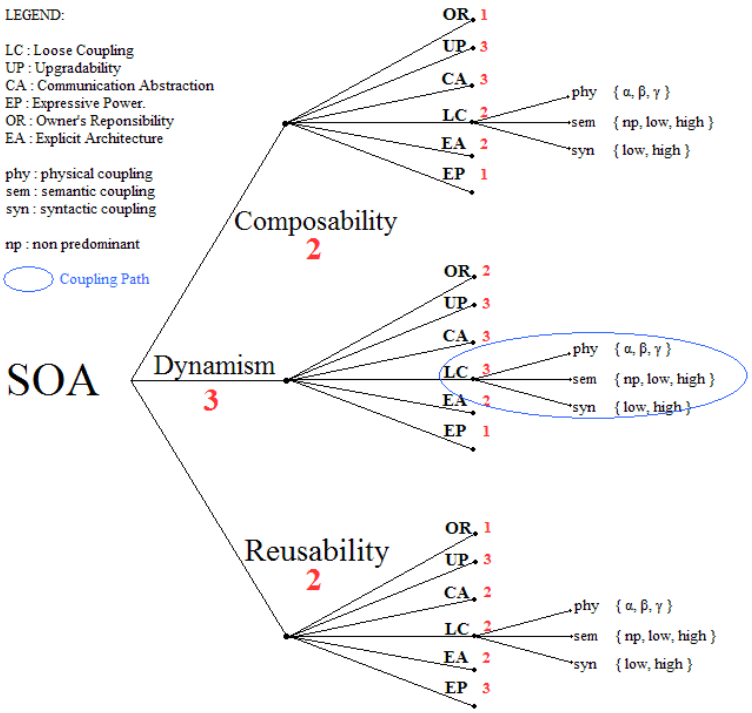

Figure 2: SOA attributes tree weighted with means of coefficients

Thus, according to the previous figure 2, we can establish that the quantitative measure of the quality of an SOA corresponds to the sum of the quality factors dynamism, reusability and composability, all three affected by their respective coefficients:

The following formula allows calculating the whole quality of an SOA.

$$SOA = 3Dynamism * 2Reusability * 2Composability$$

$$SOA = 3(2OR + 3UP + 3CA + 3(1 - LC) + 2EA + EP) * 2(OR + 3UP + 2CA + 2(1 - LC) + 2EA + 3EP) * 2(OR + 3UP + 3CA + 2(1 - LC) + 2EA EP)$$

$$SOA = 3(2OR + 3UP + 3CA + 3\,[1-(\{(a),(b),(c)\} * \left(\left(\prod_{k=1}^{\lambda}\prod_{i=1}^{N}\sum_{j=1}^{i}((P_s)_{kij})^{m_{kij}}\right) * C_{phys}\right) * P_{ndetect})\,] + 2EA + EP) *$$
$$2(OR + 3UP + 2CA + 2\,[1-(\{(a),(b),(c)\} * \left(\left(\prod_{k=1}^{\lambda}\prod_{i=1}^{N}\sum_{j=1}^{i}((P_s)_{kij})^{m_{kij}}\right) * C_{phys}\right) * P_{ndetect})\,] + 2EA + 3EP) *$$
$$2(OR + 3UP + 3CA + 2\,[1-(\{(a),(b),(c)\} * \left(\left(\prod_{k=1}^{\lambda}\prod_{i=1}^{N}\sum_{j=1}^{i}((P_s)_{kij})^{m_{kij}}\right) * C_{phys}\right) * P_{ndetect})\,] + 2EA + EP)$$

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.

V. Discussion

Because SOA implies the connectivity between several systems, commercial entities and technologies: some compromises regarding the architecture must be undertaken, and this, much more than for systems with a single application where technical difficulties prevail. 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 (for example, SAAM does not provide any clear quality metric for the architectural attributes analyzed [11]). 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 finality of our work is to design a conceptual framework and, in fine, a semi-automated prototype called SOAQE *(taking as a starting point, past methods such as ATAM or SAAM)* which could quantify with an accurate value the quality of the whole service oriented architecture.

VI. References

[1] P.S.C. Alencar, D.D. Cowan, T. Kunz and C.J.P. Lucena, "A formal architectural design patterns-based approach to software understanding," Proc. Fourth workshop on program comprehension, 2002.

[2] D. Garlan and M. Shaw, An introduction to software architecture, CMU/SEI-94-TR-21, ESC-TR-94-21, 1994.

[3] H.K. Kim, "Modeling of distributes systems with SOA and MDA," IAENG, International Journal of Computer Science, ISSN 1819-656X, Volumr: 35; Issue: 4; Start page: 509, 2008.

[4] O. Lero, P. Merson and L. Bass, "Quality attributes and service oriented architectures" SDSOA 2007, 2007.

[5] P. Bianco, R. Kotermanski and O. Merson, "Evaluating a service oriented architecture" CMU/SEI-2007-TR-015, Carnegie Mellon University, Software Engineering Institute, 2007.

[6] P. Clements, R. Kazman and M. Klein, Evaluating Software Architectures: Methods and case studied, published by Addison-Wesley Professional, 2001.

[7] T. Demarco, Controlling software projects: management, measurement and estimates, Prentice Hall, 296 pages, 1986.

[8] I. Crnkovic, M. Chaudron and S. Larsson, "Component-based development process and component lifecycle" ICSEA'06, International Conference on Software Engineering Advances, 2006.

[9] A. Hock-Koon, "Contribution à la compréhension et à la modélisation de la composition et du couplage faible de services dans les architectures orientées services" Thesis (PhD). University of Nantes, 2011.

[10] Y. Mortureux, Preliminary risk analysis. Techniques de l'ingénieur. Sécurité et gestion des risques, SE2(SE4010):SE4010.1–SE4010.10, 2002.

[11] M.T. Ionita, D.K. Hammer and H. Obbink, "Scenario-based software architecture evaluation methods: an overview", ICSE 2002, 2002.