Measuring the Degree of Service Orientation in Proprietary SOA Systems

Anwar Aldris[†], Ariadi Nugroho[*], Patricia Lago[†] and Joost Visser[*‡]
[*]Software Improvement Group, The Netherlands – {a.nugroho, j.visser}@sig.eu
[†]VU University Amsterdam – anwar.aldris@gmail.com; p.lago@vu.nl
[‡]Radboud University Nijmegen, The Netherlands

Abstract—**According to a survey conducted by Forrester Research in 2008, at least 44% of North American, European, and Asian-Pacific enterprises have adopted SOA (Service-oriented Architecture), and at least 63% would adopt it by the end of 2008 [1]. A more recent survey by Forrester also shows that SOA adoption remains strong in 2010 [2]. Nevertheless, there are many misconceptions about SOA [3], which could lead to sub optimal implementation of the paradigm. This paper is not about whether to adopt or not to adopt SOA. Instead, this paper proposes a method on how to judge an SOA implementation from architectural point of view. More specifically, we evaluate the extent to which proprietary SOA systems conform to the principles of service orientation. To this aim, a framework of Degree of Service Orientation (DoSO) has been developed. This framework is applied to nine proprietary SOA systems and the results show that, on average, the degree of service orientation is rather low. Experts' evaluation on the usefulness of the framework is also discussed.**

I. Introduction

The results of the surveys conducted by Forrester Research show an interesting phenomenon in that more and more systems are going to be built using an architectural style that puts a strong emphasis on flexibility and efficiency. However, in practice technology adoptions often do not add value to organizations. The promises and benefits of SOA have been discussed quite extensively in literature (e.g., [4], [5]) and one of the key benefits of having an SOA solution is a seamless integration of business services [6]. However, to successfully achieve such benefits organizations should understand what SOA really is and which key attributes are crucial in implementing SOA solutions.

One of the misconceptions surrounding SOA is to consider SOA merely about standards or technology [3]. Implementing an application using a web service does not guarantee an SOA, neither is implementing an Enterprise Service Bus (ESB) to support application integration. Nonetheless, the discussion concerning what it means to be SOA should be set to the original context: SOA as an architectural style.

There are known issues in SOA adoption. The most notable issue is related to performance overhead caused, for example, by exchange of messages between services that might require substantial time for parsing and validation (especially for large messages) [7]. This performance issue of SOA implementation is compounded further by network latency. Another important issue is related to configuration management of services. If configuration management of services not managed properly, evolution of services can cause instability through out the whole service chain.

The goal of this paper, however, is not to discuss the advantages or disadvantages of SOA. Instead, we aim to propose a method that can be used to judge the conformance of an SOA implementation to the principles of service orientation.

To be considered an SOA, a system should demonstrate some degree of service orientation. In other words, a system should not be considered SOA if it is not composed of services. However, different organizations might have taken different design decisions to implement SOA, which result in different degrees of service orientation.

Understanding the extent to which SOA solutions conform to the concept of service orientation is important. This is particularly true because with such knowledge we might be able to explain the success and failure of SOA implementations. Therefore, the goal of this paper is to propose, apply, and evaluate a method for measuring the degree of service orientation of SOA systems. The research questions we aim to answer are the following:

- RQ1: How can we measure the degree of service orientation of SOA systems?
- RQ2: What is the degree of service orientation in proprietary SOA systems?
- RQ3: How useful is the proposed method to characterize SOA systems from experts' point of view?

The rest of this paper is organized as follows. In Section II we introduce important concepts that will be used through out this paper. In Section III we present the answer to RQ1 by constructing a conceptual framework. Section IV discusses the design of the case studies. The answers to RQ2 and RQ3 are presented in Section IV, which discusses the results of the case studies. In Section VI we further discuss the results and limitations of our study, and finally in Section VII and VIII we discuss related work and conclusion respectively.

II. Background

A. *The Degree of Service Orientation*

In SOA context, business processes are supported by functionality that is implemented as services. These services are software components that are designed in such a way that they can support ever-changing business demands. Consequently, the degree of service orientation of an SOA system should be related to the extent to which its services are designed to be resilient towards changes in its surrounding environment.

Typically, changes in the business environment are related to changes in the partners of the business, growing service/product demands, and changes in the scope of the business. A company whose business processes rely heavily on software must ensure that when any of the aforementioned changes occurs, their software adapts to the changes easily.

Therefore, we define the degree of service orientation as the extent to which services are designed in such a way that allows them to be easily coupled, adapted and combined in order to cope with changing environment.

B. SOA Frameworks

An SOA framework or model is a set of theories or best practices that is developed to guide understanding or execution of a certain aspect or task in the area of service orientation.

Most SOA frameworks found in literature focus on the maturity of SOA adoption. These frameworks describe stages of maturity of SOA implementation and provide some key indicators at each stage. In the following passages we discuss some of these SOA frameworks.

Welke et al. proposed an SOA maturity model to guide organizations in adopting SOA by defining five maturity levels that are similar to that of CMMI (Capability Maturity Model Integration) [8]: Initial, Managed, Defined, Quantitatively Managed, and Optimized. Furthermore, the framework defines six dimensions for each maturity level: SOA view, benefits and metrics, business involvement, methodology, service sourcing, and governance. Higher maturity levels indicate a progression to a more mature use of SOA and a higher realization of SOA's ability to support the business [9]. The authors suggested that organizations could use the model to evaluate their current level of SOA maturity and then determine actions to improve the maturity to the next level. For example, to progress from the Quantitatively Managed level to the Optimized level, from the SOA View dimension an organization needs to move from having an enterprise service architecture to an adaptive architecture.

A very similar approach is proposed by Rathfelder and Groenda [10]. The authors also defined five maturity levels, but looking at five different view points: service architecture, infrastructure, enterprise structure, service development, and governance. Another similar maturity model is proposed by Arsanjani and Holley [11]. They proposed the so-called Service Integration Maturity Model (SIMM) to guide organizations transforming their applications to service-based applications. The maturity model has seven levels of maturity and seven dimensions, namely business, organization, methods, applications, architecture, information, and infrastructure.

Papazoglou proposed a slightly different maturity model. The author defined limitations of the basic SOA implementation that typically involves service requesters and service providers, and supports activities to find, bind, and publish services. The author argued that the basic SOA does not address all concerns in a service-based architecture like management, service choreography and orchestration, and security [12]. Therefore, xSOA was proposed to extend the basic SOA by defining two levels on top of the basic level, namely composition and management layers.

At the composition layer, services are aggregated to deliver new, distinct composite services to clients. At the service management layer, two functions are defined in order to manage diverse capabilities in a distributed environment, namely service operation management and open service marketplace management. Both management capabilities are expected to help manage SOA solutions in different markets.

While the aforementioned SOA frameworks generally focused on the maturity of SOA adoption, the work of Razavian and Lago proposed a framework of SOA Migration (SOA-MF) [13] [14]. Migrating legacy systems to service-oriented systems is not trivial and pose service engineering challenges. SOA-MF provides a conceptual model to understand existing SOA migration approaches in terms of supported processes (e.g., reverse engineering), required artifacts (e.g., source code, models), activities, and the exploited knowledge. Such a migration framework can help identify the strengths and weaknesses of different migration approaches.

The aforementioned frameworks and maturity models aim to provide guidelines on how to adopt or transform SOA implementation. As technology adoption inevitably requires organizational changes, many of the proposed approaches cover different issues ranging from technical to organizational. While we agree that a comprehensive view is needed in assessing SOA adoptions, more attention should be given to the underlying SOA designs. In the end, the design of an SOA implementation will determine the sustainability of the implemented solution in supporting the business goal.

III. Constructing A Framework of Degree of Service Orientation

Existing approaches that aim to characterize the nature of SOA systems (e.g., maturity) generally focus on the managerial or governance aspects. In this section, we discuss our approach to develop a framework that focuses more on internal attributes of an SOA system, namely the degree of service orientation.

A. Methodology

We follow the FCM (Factor-Criteria-Metric) approach [15] in the construction of the framework. FCM model proposed an approach that can be followed while aiming to measure any software-related attribute. The basic idea of this theory is that a software attribute can be decomposed into a set of factors and each of these factors can be also decomposed into a measurable set of criteria (sub-factors), which then can be evaluated by a set of software measurements called metrics. The relation between Factors-Criteria-Metric is shown in Figure 1.

The first step to create the framework is to identify the factors and their corresponding criteria, which is mainly based

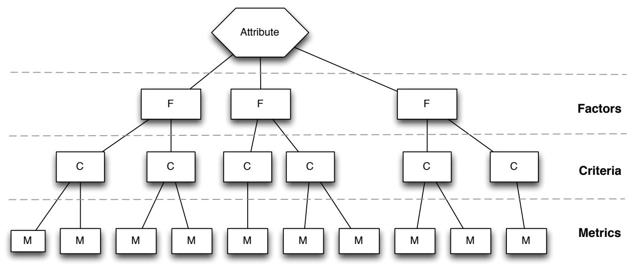

Fig. 1. Factor-Criteria-Metric Approach

on literature study. In particular, we look at existing software quality models and other related frameworks. After identifying the factors and criteria of the framework as well as the relations amongst them, we validate the framework with experts. Such a validation serve as a sanity check to make sure that the proposed framework is inline with experts' judgments. In the next sections we discuss these steps in further detail.

B. Defining Factors

The factors in the framework represent the aspects of service orientation. Recall our definition of degree of service orientation: the extent to which services are designed in such a way that allows them to easily communicate, be adapted and combined. The chosen factors should reflect these aspects.

To look for factors that reflect the aspects of service orientation we study the ISO 25010 System and Software Quality Models [16]. Figure 2 shows software product quality according to ISO 25010. The second-level entities in Figure 2 represent quality characteristics and the third-level entities represent quality properties. To be selected as a factor, a quality property in the ISO 25010 should reflect the definition of service orientation, namely it must concern the ability of a service to easily communicate, be adapted and be combined and with other services. After carefully evaluating each quality property, we find *Interoperability*, *Adaptability* and *Reusability* well suited to represent factors of service orientation.

Below are the selected factors and their definitions according to ISO 25010:

- **F.1. Interoperability**. The degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
- **F.2. Adaptability**. The degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. Adaptability includes the scalability of internal capacity (e.g., transaction volumes).
- **F.3. Reusability**. The degree to which an asset can be used in more than one system, or in building other assets.

In addition to the selected three quality properties, we considered modularity and replaceability because these quality properties seem related to the notion of service orientation. Modularity is defined as degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components [16]. Replaceability is defined as the degree to which a product can be replaced by another specified software product for the same purpose in the same environment [16].

Although modularity and replaceability seem related to the notion of service orientation, we can see that by definition both quality properties are more inward looking—that is, aimed at easing technical maintenance. Adaptability is partly about internal technical maintenance, but it also focuses on the scalability of software to process transactions requested by external parties. Given their stronger focus on internal aspects of a software system, we decided to exclude modularity and replaceability.

Having defined the factors of service orientation, the next step is to select a set of measurable criteria for each factor that satisfy its definition.

C. Defining Criteria

The factors defined previously are still in a rather abstract or conceptual level. To come up with concrete measures of the factors, we need to refine the factors further into questions that characterize how the factors can be achieved. To this aim, we basically want to be able to answer three questions: Q1) How interoperable is a service?; Q2) How reusable is a service?; and Q3) How adaptable is a service?

In the following passages we discuss the refinement of each factor into a set of concrete questions.

Q1. How interoperable is a service? From software architecture point of view, interoperability is related to interface design, layering and standardization [17]. Hence, interoperability can be broken down into fine-grained questions:

- Q1.1. Do services use a common medium to communicate and whether the messages being communicated have agreed upon syntax and semantic?
- Q1.2. Is the design of a service interface encapsulates internal details in order to increase interface stability?
- Q1.3. Is a service designed in such a way that it is not tightly coupled to other services or resources?

The above fine-grained questions are mutually exclusive and represent different aspects of interoperability: *standardization*, *abstraction*, and *loose coupling*. We will use these aspects as criteria for interoperability.

Q2. How adaptable is a service? The adaptability of a software system represents its capability to tolerate high degree of deviations in its environment [18]. For a service to be adaptable, it must have a high degree of control over its environment. Additionally, efficient request processing determines whether a service can handle substantial increase in service request. Therefore, service adaptability is related to the following two questions:

- Q2.1. Is a service designed in such a way that ensures scalability in processing requests?
- Q2.2. Is a service in control of its operating environment?

One of the most important principles of service design to achieve scalability is service *statelessness*. Managing loads of

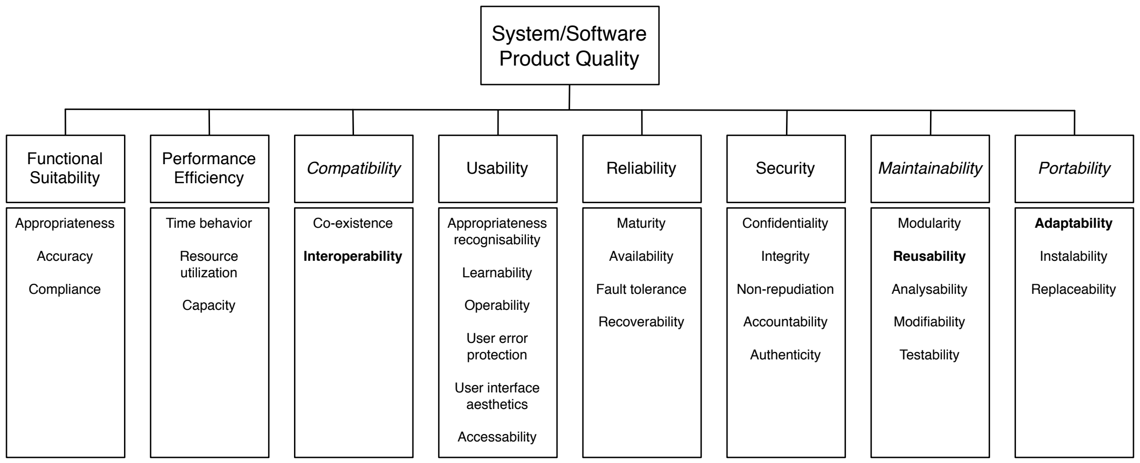

Fig. 2. ISO 25010 System/Software Product Quality

state information can compromise service performance to handle requests. We also choose service *autonomy* as a criterion of service design in order to achieve adaptable services.

Q3. How reusable is a service? Designing reusable software components typically require providing sufficient information about the component [19], making sure that the size of functionality of the component is appropriate [20] and providing a generic, extensible functionality [21]. Therefore, we can further refine the question about reusability into the following sub questions:

- Q3.1. Is the functionality of a service kept generic to promote reuse?
- Q3.2. Is a service described or populated in such a way that it is easy to find by potential consumers?
- Q3.3. Is the functionality of a service properly scoped in order to ease service composition?

Based on the above sub questions, we determine criteria that are important for reusability, namely *genericity*, *discoverability*, and *composability*.

In the following passages, we present the definitions of the aforementioned criteria.

- **C.1.1. Standardization**. The objective of standardization is to enable an efficient communication amongst services. For example this can be achieved by describing services using a standard format, invoking services through a standard communication protocols, and exposing services functionality in a standard way.
- **C.1.2. Abstraction**. Service abstraction is a design principle that imposes to publish only the required information that service consumers need in order to effectively utilize a service.
- **C.1.3. Loose coupling**. A loosely coupled system aims to isolate its underlying components. In a loosely coupled system changes in one place will not impose significant changes in other places.
- **C.2.1. Statelessness**. Service statelessness requires that services in SOA-based system not to be involved in state management tasks (e.g., keeping trace of interaction-specific or activity-specific data).
- **C.2.2. Autonomy**. Service autonomy is a design principle that empowers service control over their execution environment, which increases the reliability of the service.
- **C.3.1. Genericity**. A service is considered generic if the functionality that the service provides could be reused in different contexts.
- **C.3.2. Discoverability**. Discoverability is the ability of a service to be discovered by supplementing services with communicative meta data by which they can be effectively discovered and interpreted [22].
- **C.3.3. Composability**. Service composability represents the extent to which a service, together with other services, can be composed to deliver a new service.

In line with the aforementioned criteria, principles of service design have been proposed in the literature. Papazoglou, for instance, argued that services should be technology neutral, loosely coupled, and support location transparency in order to be consumable by various parties from within or beyond an enterprise boundary [23]. The author also considered self-containment (service maintains its own state), dynamic discovery, dynamic invocation, and dynamic (re-)composition essential properties of services in SOA systems [24].

D. Defining Metrics

Metrics are the measurement used to evaluate the proposed criteria. Metrics often are measured from software artifacts automatically or semi-automatically. We consider automated static code analysis as the more objective and reliable approach to gather measurement data. However, not all criteria defined in Section III-C can be measured from software artefacts. Composability, for example, is difficult to assess without manually inspecting the design and scope of services.

Considering the needs for expert judgments and domain knowledge in order to evaluate services based on the defined criteria, we use qualitative assessment (i.e., using expert opinions) as metrics. In section IV-B, we further discuss the

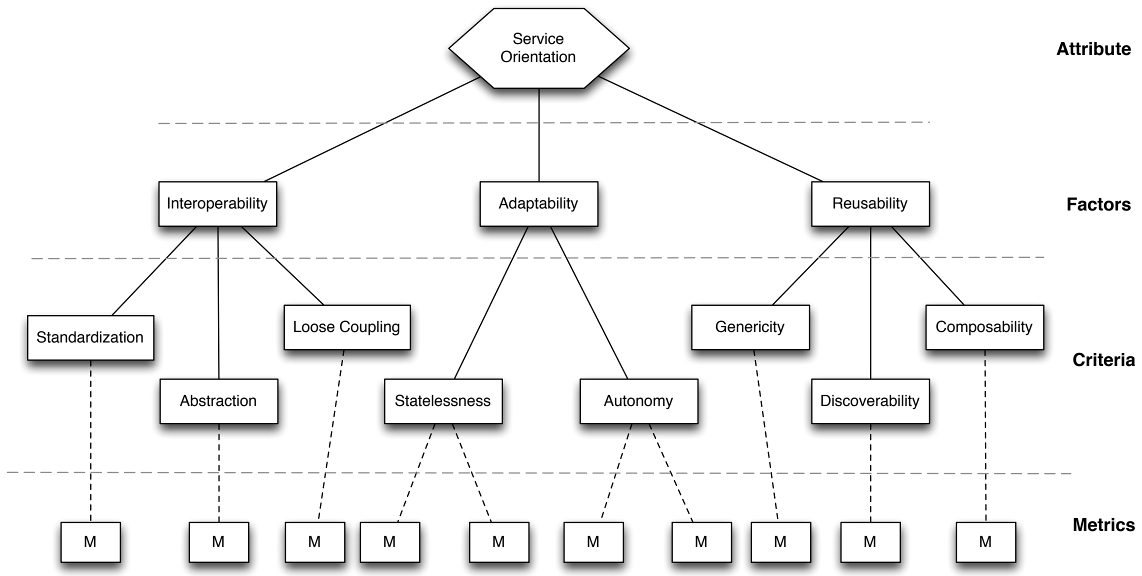

Fig. 3. The Framework of Degree of Service Orientation

instruments used to perform the qualitative assessment.

E. The DoSO Framework

Following the FCM approach, our goal is to be able to measure the degree of service orientation of an SOA system. The degree of service orientation, as discussed earlier in Section II-A, can be decomposed into three main factors, namely interoperability, adaptability, and reusability. In order to measure interoperability, adaptability, and reusability, we consider the eight criteria presented earlier. We found that the eight criteria are conceptually related to the notions of interoperability, adaptability, and reusability.

Figure 3 presents the proposed framework for evaluating the degree of service orientation of SOA systems. Notice that the framework has four levels, namely attribute, factors, criteria, and metrics. The figure shows the mapping between the eight criteria and the higher level factors, namely interoperability, adaptability, and reusability. The mapping indicates the most significant relationships between the factors and the criteria. For example, abstraction, loose coupling, and standardization are believed to have the most significant influence on interoperability.

The relations between Criteria and Metrics are represented as dashed lines because the metrics defined are context dependent. Metrics are the results of operationalizing the concepts of interest (e.g., abstraction), which formulation depends on the available artifact from which it is going to be measured. Therefore there is no dedicated metrics defined to measure the criteria. In the next section the operationalization of the framework will be discussed.

In the following passages we discuss the rationale behind the mapping between criteria and factors of the framework.

Interoperability emphasizes the ability of different services to work together. From technical point of view, interoperability means that the exchange of information between services can take place without burden. Standardization supports interoperability because the use of common standards (e.g., standard communication protocols) will ease data exchange. Abstraction (information hiding/encapsulation) hides underlying service details from consumers in order to provide stable interfaces. Poor abstraction also directly affects loose coupling because it increases the probability of the implementing components to change. Therefore, both abstraction and loose coupling support interoperability by ensuring sustainable exchange of information that is achieved through having stable interfaces.

Adaptability concerns the ability of a service to handle a growing number of users or consumers or its ability to be easily rescaled (in terms of resources) to respond to such growth. A service with a high degree of autonomy will have full control over its operating environment and resources. As a result, rescaling its resources is easy because they are not supplied by other parties. Concerning statelessness, a service with high degree of statelessness do not need to allocate resources for managing state information, which increases its ability to serve more requests concurrently.

Reusability represents the capability of a service to be exploited and adapted in such a way that it can support new business processes. An SOA system with generic services is cheaper and easier to extend because generic functionality can be reused elsewhere. With composable services one is allowed to construct new composite services to cater new business processes, hence facilitates extensions of the business scope. Finally, discoverability facilitates reusability in that it increases the degree of exposure of a service to the outer world. This exposure increases the chance of a service to be used in different business contexts.

With the proposed DoSO framework we aim to assess the

TABLE I
CHARACTERISTICS OF THE CASE STUDIES

System	Industry Sector	Main Technology	Size (KLOC)
System A	Insurance	C#	76
System B	Government	PHP	96
System C	Government	Java	118
System D	Healthcare	C#	153
System E	Government	Java	154
System F	Transport	Java	207
System G	Consumer goods	VB.NET	459
System H	Insurance	Java	1,170
System I	Government	Java	2,095

degree of service orientation of SOA systems, as presented in the next section. The factors of the framework are based on the ISO 25001, and they are further decomposed into a measurable set of criteria. The proposed DoSO framework gives an answer to RQ1: how can we measure the degree of service orientation of SOA systems?

IV. CASE STUDIES

The objective of the case studies is to apply and evaluate the usefulness of the proposed DoSO framework. We aim to answer RQ2 and RQ3:

- RQ2: What is the degree of service orientation in proprietary SOA systems?
- RQ3: How useful is the framework to characterize SOA systems from experts' point of view?

The case studies are performed within the contexts of systems that are/were assessed by the Software Improvement Group (SIG). SIG is an Amsterdam-based consultancy firm that provides services to assess, monitor and, together with TÜViT, certify the technical quality of software [25].

A. System Selection

SIG maintains a software repository that currently stores around 700 software systems. As we are only interested in SOA systems, we first need to exclude systems that are not designed based on services.

Web service is the most common implementation of SOA systems in practice. Web Service Definition, specified using Web Service Language (WSDL), is one important element of web service implementation. Therefore, to identify potential SOA systems we look for WSDL files in the source code directories of every system. This identification process is done automatically using a script, and it resulted in 47 systems being identified using web service technology.

The fact that a system is built using web services does not necessarily mean it is an SOA system. Therefore, we further clarify whether the identified systems are indeed SOA systems by interviewing SIG consultants who are familiar with the systems. We asked the consultants whether the systems meet the basic characteristic of an SOA, namely whether it is composed of services. Additionally, only fully functional and independent systems were considered—for example, we excluded systems that never go live or subsystems that can not operate independently.

As there is a possibility that SOA systems' interfaces are not implemented using WSDL, we also asked the consultants to nominate other SOA systems that were not in our list. In the end, we identified nine SOA systems to be used in the analysis. Table I provides a summary of characteristics of the nine systems.

B. Instruments to Collect Data

The proposed DoSO framework is designed as a generic framework that can be used in different contexts. From the point of view of measurements, different contexts often mean different data collection strategies.

Three data sources were considered during the operationalization of the framework, namely source code, recovered software architecture, and expert opinions. Static code analysis and analysis of recovered architecture are considered as the most objective and reliable source of data. However, as mentioned earlier, not all the criteria defined can be measured automatically from software artifacts. These criteria such as composability and autonomy require qualitative assessments or expert judgements.

Therefore, the use of qualitative metrics based on expert opinions is considered a more realistic way to measure the proposed criteria. The use of expert judgement in software engineering is not uncommon (see for example in [26] and [27]). Furthermore, the experts selected in the case studies will have full access to systems' source code and other system documentations, which should increase the reliability of the information that they provided.

In the following passages we discuss the operationalization of the DoSO framework in terms of questionnaires to collect data from experts.

1) DoSO Assessment Questionnaire: An assessment questionnaire is used to collect expert opinions concerning the degree of service orientation. For every criteria in the DoSO framework (e.g., abstraction), we defined questions measured in a Likert scale with the following items: very low degree - low degree - average - high degree - very high degree. Table II shows the questions and their mappings to the criteria defined in the framework.

As shown in Table II, there are 15 questions in the questionnaire. However, some questions are optional (i.e., M.3.2.3 - M.3.2.5) depending on the answer given to the preceding questions (i.e., M.3.2.1 and M.3.2.2).

2) Feedback Questionnaire: A feedback questionnaire is used as an instrument to capture experts' evaluation of the DoSO framework. Three aspects were evaluated using the questionnaire, namely experts' acceptance of the framework, the perceived effectiveness of the framework, and the completeness of the framework.

To evaluate experts' acceptance, we use method adoption criteria discussed in literature (e.g., [28]) as shown in Table III. Additionally, to assess the effectiveness and completeness of the framework, we ask the experts to rate the extent to which the framework is able to measure the degree of service orientation and to name important aspects currently missing in

TABLE II
QUESTIONS THAT SERVE AS QUALITATIVE METRICS AND THEIR MAPPINGS TO THE MEASURED CRITERIA.

Questions	**Criteria**
M.1.1.1: To what degree services use the same communication protocol(s)? M.1.1.2: To what degree services need data format/type transformation before being able to communicate?	C.1.1. Standardization
M.1.2.1: To what degree service interfaces hide the details of the implementation?	C.1.2. Abstraction
M.1.3.1: To what degree service implementations have minimal, but well-known dependencies on the services they invoke?	C.1.3. Loose coupling
M.2.1.1: To what degree services follow asynchronous communication? M.2.1.2: To what degree services avoid keeping state information?	C.2.1. Statelessness
M.2.2.1: To what degree services' life cycle (e.g. deployment and maintenance) is independent of that of other services? M.2.2.2: To what degree services have control over underlying runtime execution environment?	C.2.2. Autonomy
M.3.1.1: To what degree services functionality is generic enough to be used in different domains (e.g.: marketing, sales)?	C.3.1. Genericity
M.3.2.1: Does the system use service registry to publish any of its services? M.3.2.2: Does the organization have its own service registry? M.3.2.3*: To what degree service registry supports management of names and locations of services? M.3.2.4*: To what degree service registry supports management of registration and querying of services? M.3.2.5*: To what degree service registry supports dynamic service matching?	C.3.2. Discoverability
M.3.3.1: To what degree services are coarse-grained services?	C.3.3. Composability

* Only asked if the experts answer YES to M.3.2.1.

the framework. Except for the question that asked the experts to name missing aspects in the framework, all questions are measured in Likert scale, with items similar to those of the DoSO assessment questionnaire.

C. Execution

Prior to the execution of the case studies, we approached consultants who are/were involved in the technical quality assessments of the selected systems. Initially, we aimed to use distinct consultants for each of the nine systems. However, because two of the consultants were familiar with some systems, they were asked to assess more than one system (maximal two systems). Hence, in the end there were seven consultants participated in the case studies. In the rest of this paper we refer to the consultants as experts.

The nine case studies were executed in a more or less parallel manner depending on the availability of the experts. The execution of a case study consisted of three separate sessions, namely the introduction session, assessment session, and feedback session.

In the introduction session, we briefed the experts about the task that they need to perform in the assessment session. Together with the experts, we went through the assessment questionnaire and provided clarification as needed. The introduction session on average lasted approximately 30 minutes.

The assessment session is an offline session—that is, the experts performed an assessment of an SOA system without assistance. In the session, the experts were asked to complete the assessment questionnaire. Additionally, they were asked to answer extra questions about system characteristics and respondent background. The experts were given several days to finish this task because of their busy schedules and the necessity to look at system source code and documentations. On average, the experts returned the completed questionnaire within a week.

After receiving the completed assessment questionnaire of a system, the feedback session would then take place. In this session we provided the result of the analysis, namely the degree of service orientation according to the DoSO framework, to the experts. After explaining the results and possibly with some open discussions on the results, in the end we asked the experts to complete the feedback questionnaire.

V. RESULTS

In this section we discuss the findings of the case studies. We start by discussing the results of assessing nine SOA systems using the DoSO framework. Subsequently, we discuss experts' evaluation on the usefulness of the framework.

A. Assessments of Degree of Service Orientation

Table IV shows the results of applying the DoSO framework to the case studies. The columns in the table represent elements in the three levels of the framework, namely criteria, factor, and goal (measured attribute: DoSO). The scores in the criteria level represent scores associated with each Likert item: 1 - very low degree; 2 - low degree; 3 - average; 4 - high degree; 5 - very high degree. Scores of criteria that are composed of

TABLE III
FRAMEWORK VALIDATION CRITERIA

Criteria	Definitions
Perceived Usefulness	The extent to which users believe that a method improve their job performance
Perceived Ease of Use	The extent to which users believe that a method can be utilized without much effort (hassle free)
Compatibility	The extent to which a method is compatible with existing norms or past experiences of potential users
Subjective Norm	The extent to which users think that people important to them (e.g., colleagues, mentors, managers) would encourage them to adopt a method
Voluntariness	The extent to which users think that they will adopt a method voluntarily

more than one questions (e.g., standardization) are calculated based on the median of the scores of the composing questions. Similarly, scores in the factor level are the medians of the composing criteria. The highest level of the model (i.e., DoSO), however, is based on the geometric mean of the three factors. Geometric mean is an aggregation method that ensures no variable dominates the result of the aggregation [29].

In Table IV we can see that System A has the highest degree of service orientation (3.6) and System E the lowest. System A scores above average in all criteria except for the degree of autonomy and composability. However, on average, System A scores from medium to high for interoperability, adaptability, and reusability. This results in high degree of service orientation for System A.

System E, on the other hand, scores below average for all criteria except standardization and autonomy. This results lead to low degree of interoperability, adaptability, and reusability, and eventually low degree in service orientation. Another point to notice from the table is that most systems (seven of them) do not publish their services in a service registry. System F uses a service registry, but the corresponding experts are not aware of the details of the registry. Therefore, for System F discoverability is not taken into account for calculating the reusability score.

By looking at the results of the assessments in Table IV we can quickly find weak spots related to service orientation in a system. Moreover, we can elaborate the analysis by looking at the exact question that scores low and perform deeper analysis in the corresponding system for verification. For example, if the degree of service abstraction is low, we can investigate further whether there were conscious decisions that justify such a design decision. Otherwise, this finding can be used to identify future improvements.

So far we have discussed how we can use the results of the assessment to perform analysis per system. Looking at the overall results across systems can help reveal general trends concerning the degree of service orientation. In Figure 4, the summary of the criteria scores is shown using boxplots. Bold horizontal lines in boxes represent median values. From Figure 4 we learn that the nine systems, on average, score high in standardization and statelessness, medium in abstraction and loose coupling, and low in autonomy, genericity, discoverability, and composability.

The summary of the scores at the factor and goal levels is shown in Figure 5. The figure shows that the systems, on average, score medium in interoperability (composed of

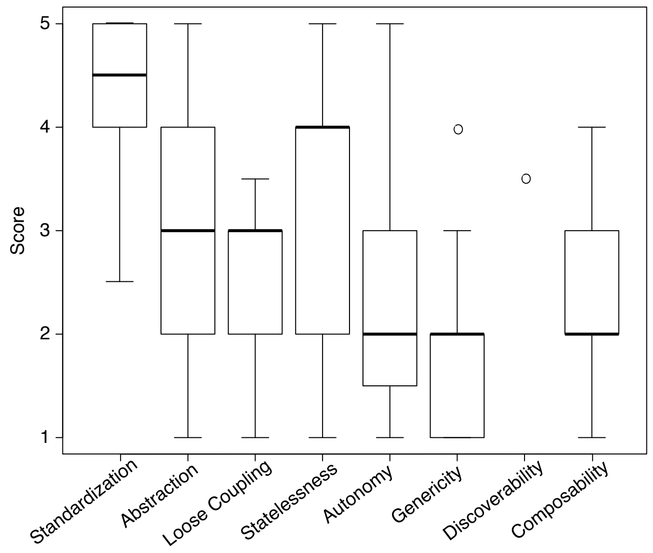

Fig. 4. Summary of scores across criteria

standardization, abstraction, and loose coupling) and slightly below medium for adaptability (composed of statelessness and autonomy). Furthermore, on average the nine systems score low in reusability (composed of genericity, discoverability, and composability). The fact that on average the systems score low in reusability can be explained by the low scores in its composing criteria: genericity, discoverability, and composability—as shown in Figure 4.

Finally, the degree of service orientation is shown by the rightmost boxplot in Figure 5. None of the systems scores above 4.0 (high degree of service orientation), and half of the systems score between 1.8 to 3.2. Nevertheless, on average, the degree of service orientation of the nine systems is rather low (scores 2.4 out of 5.0). This result gives answer to the second research question (RQ2): What is the degree of service orientation in proprietary SOA systems?

B. Experts' Evaluation of the DoSO Framework

In this section we present the results of experts' evaluation of the DoSO framework after using it to assess SOA systems. The evaluation criteria, as mentioned earlier, are perceived usefulness, perceived ease of use, compatibility, subjective norm, and voluntariness (see Table III).

Most of the experts who participated in the case studies stated that they have been engaged at least in one assessment

TABLE IV
OVERALL SCORES OF DOSO ASSESSMENT

System	Standardization	Abstraction	Loose Coupling	Statelessness	Autonomy	Genericity	Discoverability	Composability	Interoperability	Adaptability	Reusability	Service Orientation
System A	4.0	5.0	3.0	4.0	2.5	4.0	3.5	2.0	4.0	3.3	3.5	3.6
System B	2.5	3.0	2.5	4.0	1.5	2.0	0.0	2.0	2.5	2.8	2.0	2.4
System C	3.5	2.0	2.0	4.0	2.0	1.0	0.0	3.0	2.0	3.0	1.0	1.8
System D	5.0	3.0	3.0	1.0	1.0	2.0	0.0	2.0	3.0	1.0	2.0	1.8
System E	5.0	1.0	1.0	1.0	2.0	1.0	0.0	1.0	1.0	1.5	1.0	1.1
System F	5.0	4.0	3.5	2.0	3.0	1.0	NA	3.0	4.0	2.5	2.0	2.7
System G	4.5	1.0	1.0	5.0	1.0	1.0	0.0	2.0	1.0	3.0	1.0	1.4
System H	5.0	4.0	3.5	4.0	5.0	2.0	0.0	4.0	4.0	4.5	2.0	3.3
System I	4.5	5.0	3.0	2.0	3.0	3.0	0.0	3.0	4.5	2.5	3.0	3.2

NA represents a missing value; hence is not taken into account in the aggregation.

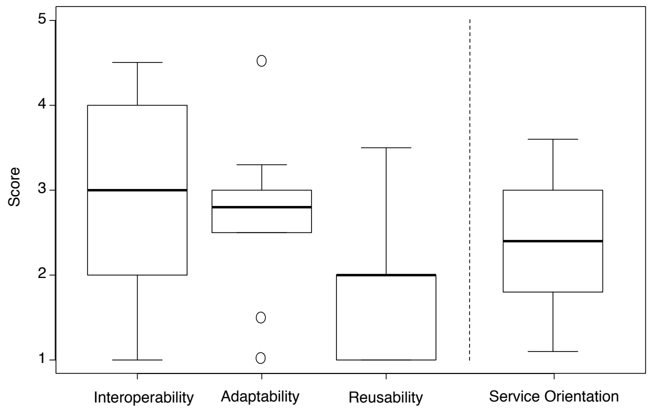

Fig. 5. Summary of scores of interoperability, adaptability, reusability, and service orientation

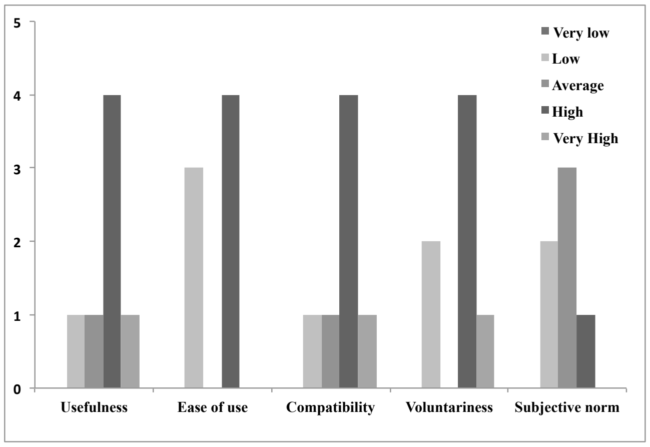

Fig. 6. Results of framework adoption evaluation

of SOA system in the last ten years. Furthermore, two experts hold PhD degree and the rest hold master degree in computer science or closely related subject.

Figure 6 shows the results of experts' opinions in terms of the aforementioned criteria. The chart represents the opinions of the seven experts, except for the subjective norm criteria for which only six experts responded. In the figure we can see that the majority of the experts rated high for the degree of usefulness, ease of use, compatibility and voluntariness of the DoSO framework. Furthermore, we see that there is a polarized opinion for the ease of use of the framework—that is, four experts rated high degree and three rated low degree. Finally, many of the experts seemed undecided when asked about the degree to which other people would use the framework—five experts rated low to average degree, while one rated high degree.

Additionally, we asked the experts to rate the degree to which the framework is able to measure the degree of service orientation. Six experts rated *at least* high degree, while only one rated low degree. This result indicates that most of the experts agreed with the assessment results given by the DoSO framework. Moreover, when asked about the completeness of the framework, five experts indicated that there is no missing aspect in the framework. For the rest, one expert did not answer and the other one indicated that what missing is a scoping parameter—that is, the framework should be applied to comparable types of services (e.g., data services), rather than to services of different types.

To understand the experts' rationale, we look further at their textual comments. We identify four important points as follows. First, the DoSO framework is considered useful because it provides common terminology when evaluating SOA systems. Second, while the majority of the experts considered the framework easy to use, some stated the opposite. Those who doubted the ease of use of the framework argued that it took some effort to collect all the required information, and in real assessments they might need to approach the clients to collect the information. Third, the framework has a high compatibility with the experts' experience because it is structured in the same way as the quality model for software maintainability used by SIG. Finally, concerning the subjective norm, many experts considered the framework too specific (e.g., applicable only to SOA systems) and it requires further validations before it can be used by a broader audience.

All in all, from the experts' evaluation we see some positive feedback concerning the usefulness and compatibility of the framework. However, we also see that automation of data collection and further validation of the framework are considered as important aspects to be improved in order to increase the chance of future adoption. These results give answer to RQ3: How useful is the framework to characterize SOA systems from experts' point of view?

VI. Discussion

In the previous sections we have discussed our approach to define a framework to assess the degree of service orientation of SOA systems, its applications to proprietary SOA systems, and the results of such applications. In this section we further discuss the findings.

A. Interpretation of Results

From the applications of the DoSO framework, we learn the following:

- The DoSO framework helps identify weak aspects related to service orientation in SOA systems.
- Within the contexts of the case studies, the degree of service orientation of SOA systems is rather low.
- The reusability of the SOA systems is found to be relatively weaker compared to their interoperability or adaptability.
- The majority of the experts who used the DoSO framework consider it highly useful, easy to use, and compatible with their experience. Moreover, they would highly consider using the framework in the future.

There is a few points that we need to elaborate further concerning the aforementioned findings.

The DoSO framework is a generic framework that can be applied in different contexts using different sources of data depending on available artifacts. In this study we use expert opinions to provide input to the model and we found that the framework is able to capture different degrees of service orientation in the case studies. In other contexts, one can opt for defining more objective metrics to operationalize the framework. Regardless of its operationalization, however, the framework provides common terms or key indicators for evaluating SOA systems.

In the context of the nine studied systems, we have found that according to the experts the average degree of service orientation of these systems is rather low (slightly below average). Amongst all the criteria, only service standardization and statelessness, on average, score above average. These results might reflect that some principles of SOA design are still not considered important or there is little awareness about some principles of SOA design.

The fact that the SOA systems are generally poor in terms of reusability might indicate an interesting phenomenon: services are rarely designed to be highly reusable or composable, let alone publishable. This result is not necessarily bad. Instead, it may reflect the types of problems that typically drive the development of SOA solutions. For example, out of the nine systems, only one system (System A) scores above average for reusability. This system is a custom-made SOA solution that handles various insurance handling processes in the front- and mid office of an insurance firm. It is very likely that services in this system were designed to support reusability because the firm anticipates that more and more services will be delivered by the system. Therefore, highly extendable system is seen as a prerequisite to facilitate such a demand. In the other systems, integration issues may have been the main driver for introducing SOA solutions, and hence there was no incentive to design services that are highly reusable, composable, or discoverable. Nevertheless, further investigation is needed to support this claim.

One important lesson that we also learn from the case studies is that the framework can help frame the discussion about expectations from an SOA solution. Depending on the needs of the business, the initial SOA adoption can be focused on achieving a certain factor such as interoperability. Subsequently, future evolution of the system can be driven by the necessity to achieve the other factors or to improve the already achieved factor even more. This can be formulated in terms of an adoption/evolution roadmap that is based on clear and measurable goals.

B. Limitations

We identified several limitations of this study as follows.

- *Framework completeness*
  In this study, we have developed a framework to assess the degree of service orientation based on the ISO 25010. Although we believe the framework covers most of the significant aspects of service orientation, it is not necessarily complete. To assess its completeness, we have performed validations with seven experts. However, experts did not see major aspects missing in the framework.
- *Subjective assessments*
  The application of the DoSO framework in the case studies is based on expert judgements, which rely on experts' knowledge about the system and data they have collected in the assessment projects. To increase the reliability of their judgements, the experts were given sufficient time to look up the required artifacts and to complete the tasks of this study.
- *Generalizability*
  The systems used as case studies are proprietary systems of different sizes and types, and are currently operational. To some extent, this guarantees the representativeness of the systems in the sample. Nevertheless, we have to be careful in generalizing the results of this study because it is only based on the assessments of a few SOA systems. Additionally, experts involved in this study are SIG consultants who are already familiar with software assessment based on ISO-based quality models. Therefore, we cannot claim that software engineers beyond the

context of this study will be in agreement with the results concerning the adoption level of the framework.

VII. Related Work

To the best of our knowledge there is little research performed to investigate the degree of service orientation of proprietary SOA systems. A closely related work was done by Shim et al., which proposed a design quality model for service-oriented architecture [30]. The authors defined four quality attributes to represent quality, namely effectiveness, understandability, flexibility, and reusability. These quality attributes are mapped to seven design properties including coupling, cohesion, and complexity, and a set of design metrics is defined to measure these design properties. The quality model was applied to two versions of a research management system and the authors claimed that the changes in quality conformed to their expectation. However, no formal validation was done to support such a claim.

A study similar to the work of Shim et al. was conducted by Alahmari et al. The authors focused on service granularity as quality attribute of SOA and proposed a set of metrics to measure it [31]. This set of metrics is composed of data, functionality and operations granularity metrics. The authors further assessed the impacts service operations granularity on other service attributes, namely service operations complexity, cohesion and coupling, in five refactoring scenarios. The result showed that service operations granularity potentially affects service operations complexity to a high degree.

Bianco et al. established a collection of design-related questions to evaluate SOA [32]. These SOA design questions include aspects such as service granularity, service security, and the use of service registry. The authors also provided an example of applying the design questions to evaluate an SOA system using the Architecture Trade-off Analysis Method (ATAM) [33].

The work of Hau et al. [34] identified contexts in which SOA design principles would be most beneficial. For example, interface orientation (abstraction) is considered helpful in the following circumstances: big and complex projects, projects with multiple teams, projects with volatile requirements and time constraints. The authors considered the contexts as criteria to assess how SOA should be adopted in organizations.

VIII. Conclusion and Future Work

In this paper we report our investigation on measuring the degree of service orientation of SOA systems. A framework of degree of service orientation (DoSO framework) is proposed and applied to nine case studies involving nine proprietary SOA systems and seven experts who are familiar with the systems. Below we summarize the answers to the research questions.

- *RQ1: How can we measure the degree of service orientation of SOA systems?*
  To measure the degree of service orientation we have developed a conceptual framework based on the ISO 25010. The factors of the framework are interoperability, adaptability and reusability. These factors are decomposed further into measurable criteria such as standardization, loose coupling and compossability. This so-called DoSO framework is operationalized using qualitative metrics based on expert judgments and are applied to case studies of nine SOA systems.
- *RQ2: What is the degree of service orientation in proprietary SOA systems?*
  Based on the results of nine case studies, we learn that, on average, the degree of service orientation of the analyzed SOA systems is rather low (scores 2.4 out of 5.0). Poor reusability of services (determined by genericity, composability, and discoverability of services) mainly explains the low degree of service orientation.
- *RQ3: How useful is the framework to characterize SOA systems from experts' point of view?*
  The majority of experts who used the DoSO framework to assess the nine SOA systems consider it useful, easy to use, compatible with their experience, and has the potential for future adoption.

Future work should aim to operationalize the DoSO framework based on a more objective measurement data such as code metrics or design/architectural metrics. Defining such metrics and extracting them using an automated tool will greatly increase the usefulness of the framework. Additionally, further research should evaluate the completeness of the aspects covered in the framework. Finally, to further assess the usefulness and practicality of the framework, more case studies in different contexts are needed.

References

[1] R. Heffner, "SOA adoption: Budgets don't matter much," *Forrester Research*, 2008.

[2] R. Heffner, G. Leganza, and L. Blackburn, "SOA adoption 2010: Still important, still strong," Forrester Research, Tech. Rep., 2011.

[3] G. Lewis, E. Morris, S. Simanta, and L. Wrage, "Common misconceptions about service-oriented architecture," in *Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007. ICCBSS'07. Sixth International IEEE Conference on.* IEEE, 2007, pp. 123–130.

[4] Z. Mahmood, "The promise and limitations of service oriented architecture," *International Journal of Computers*, vol. 1, no. 3, pp. 74–78, 2007.

[5] M. Huhns and M. Singh, "Service-oriented computing: Key concepts and principles," *Internet Computing, IEEE*, vol. 9, no. 1, pp. 75–81, 2005.

[6] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham, "Impact of service orientation at the business level," *IBM Systems Journal*, vol. 44, no. 4, pp. 653–668, 2005.

[7] L. O'Brien, P. Brebner, and J. Gray, "Business transformation to soa: aspects of the migration and performance and qos issues," in *Proceedings of the 2nd international workshop on Systems development in SOA environments.* ACM, 2008, pp. 35–40.

[8] CMMI Product Team, "Capability maturity model® integration (CMMI SM), version 1.1," Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA, Tech. Rep. SEI-2002-TR-012, 2002.

[9] R. Welke, R. Hirschheim, and A. Schwarz, "Service-oriented architecture maturity," *Computer*, vol. 44, no. 2, pp. 61–67, 2011.

[10] C. Rathfelder and H. Groenda, "isoamm: an independent soa maturity model," in *Distributed Applications and Interoperable Systems.* Springer, 2008, pp. 1–15.

[11] A. Arsanjani and K. Holley, “The service integration maturity model: achieving flexibility in the transformation to soa,” in *Services Computing, 2006. SCC’06. IEEE International Conference on*. IEEE, 2006, pp. 515–515.

[12] M. Papazoglou, “Extending the service oriented architecture,” *Business integration journal*, vol. 7, no. 1, pp. 18–21, 2005.

[13] M. Razavian and P. Lago, “A frame of reference for SOA migration,” in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, E. Di Nitto and R. Yahyapour, Eds. Springer Berlin / Heidelberg, dec 2010, vol. 6481, pp. 150–162.

[14] ——, “A survey of SOA migration in industry,” in *Service Oriented Computing*, ser. LNCS, vol. 7084. Berlin: Springer-Verlag, dec 2011, pp. 618–626.

[15] J. McCall, P. Richards, and G. Walters, “Factors in software quality - concepts and definitions of software quality,” DTIC Document, Tech. Rep., 1977.

[16] ISO, “ISO/IEC 25010: 2011, Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-System and software quality models,” 2011.

[17] M. Kasunic, “Measuring systems interoperability: Challenges and opportunities,” Software Engineering Institute, Tech. Rep., 2001.

[18] N. Subramanian and L. Chung, “Metrics for software adaptability,” *Proc. Software Quality Management (SQM 2001), April*, 2001.

[19] H. Washizaki, H. Yamamoto, and Y. Fukazawa, “A metrics suite for measuring reusability of software components,” in *Software Metrics Symposium, 2003. Proceedings. Ninth International*. IEEE, 2003, pp. 211–223.

[20] R. Haesen, M. Snoeck, W. Lemahieu, and S. Poelmans, “On the definition of service granularity and its architectural impact,” in *Advanced Information Systems Engineering*. Springer, 2008, pp. 375–389.

[21] M. Price and S. Demurjian Sr, “Analyzing and measuring reusability in object-oriented design,” in *ACM SIGPLAN Notices*, vol. 32, no. 10. ACM, 1997, pp. 22–33.

[22] T. Erl, *SOA: Principles of Service Design*. Prentice Hall Press, 2007.

[23] M. Papazoglou, “Service-oriented computing: Concepts, characteristics and directions,” in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*. IEEE, 2003, pp. 3–12.

[24] M. Papazoglou and W. Van Den Heuvel, “Service oriented architectures: approaches, technologies and research issues,” *The VLDB journal*, vol. 16, no. 3, pp. 389–415, 2007.

[25] R. Baggen, K. Schill, and J. Visser, “Standardized code quality benchmarking for improving software maintainability,” in *Proceedings of the 4th International Workshop on Software Quality and Maintainability*, 2010.

[26] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *Journal of Systems and Software*, vol. 70, no. 1, pp. 37–60, 2004.

[27] M. Höst and C. Wohlin, “An experimental study of individual subjective effort estimation and combinations of the estimates,” in *Proceedings of the 20th international conference on Software engineering*. IEEE Computer Society, 1998, pp. 332–339.

[28] C. Riemenschneider, B. Hardgrave, and F. Davis, “Explaining software developer acceptance of methodologies: a comparison of five theoretical models,” *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1135–1145, 2002.

[29] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation functions: a guide for practitioners*. Springer Publishing Company, Incorporated, 2008.

[30] B. Shim, S. Choue, S. Kim, and S. Park, “A design quality model for service-oriented architecture,” in *Software Engineering Conference, 2008. APSEC’08. 15th Asia-Pacific*. Ieee, 2008, pp. 403–410.

[31] S. Alahmari, E. Zaluska, and D. De Roure, “A metrics framework for evaluating soa service granularity,” in *Services Computing (SCC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 512–519.

[32] P. Bianco, R. Kotermanski, and P. Merson, “Evaluating a service-oriented architecture,” Software Engineering Institute, Tech. Rep., 2007.

[33] R. Kazman, M. Klein, and P. Clements, “ATAM: Method for architecture evaluation,” Software Engineering Institute, Tech. Rep., 2000.

[34] T. Hau, N. Ebert, A. Hochstein, and W. Brenner, “Where to start with soa: Criteria for selecting soa projects,” in *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*. IEEE, 2008, pp. 314–314.