Supporting Service-Oriented Design with Metrics

Helge Hofmeister and Guido Wirtz
Distributed and Mobile Systems Group, Otto-Friedrich Universität, Bamberg, Germany
Email: hofmeister@ecoware.de, guido.wirtz@uni-bamberg.de

Abstract

The service-oriented architectural style is widely perceived today. However, service orientation is a very general concept and its application in real-life situations is somewhat ambiguous. This is partially due to the fact that service-oriented principles are subject to interpretation rather than elements of the style. In this paper we propose a set of design metrics for service-oriented design. Based on an established metric for the coupling of component-based systems we introduce a complexity metric by the means of service coupling. We argue that service aggregators and the centralization of a system's control flow is appropriate to address a system's complexity. In order to approach an objective design that incorporates these principles, we introduce four metrics for the analysis of how a system handles its complexity with service-oriented means. Finally, we apply the presented metrics to an enterprise-scale real-life case study that we have conducted.

Keywords: SOA, metrics, control centralization, service aggregation, modifiability

1 Introduction

The service-oriented architectural style allows to build applications that reuse distributed functionality of heterogeneous application landscapes. It is an evolution of component-based approaches that emphasis "soft" design principles (loose coupling, abstraction, autonomy – cf. [6]). Applications that reuse functionality and expose their functionality as web-based applications are so-called composite applications [15]. Together, service orientation (SO) and composite applications promise to protect the investment into legacy landscapes by reusing the existent functionality while allowing for adoption of business changes. This is why SO is an approach that allows large organizations to increase the modifiability of their application landscape and especially of the business processes that are implemented. However, when it comes to mid- to large-scale projects that are conducted in large organizations, it becomes crucial that the design of composite applications is not only conducted and understood by a small group of specialists.

In this paper we propose a set of metrics that objectively capture some important principles of SO and can be applied throughout projects. By using (and refining) these metrics, organizations may build up a common understanding of SO that helps both, to improve the overall attainability of service-oriented principles and to communicate the differences of SO in comparison to more traditional approaches. Hence, the application of the presented metrics supports the proper realization of service-oriented architectures (SOA).
After presenting related work in section 2, we shortly introduce in section 3 the concepts of service aggregation and control centralization as mechanisms to increase the modifiability of a system. The metrics that incorporate these ideas are introduced and discussed in section 4. The interpretation of the metrics in the context of an enterprise-scale case study is outlined in section 5. The paper concludes with an outlook on future work in section 6.

2 Related Work

Starting in the mid-seventies and lasting till the mid-nineties, software quality metrics were intensively researched. In order to increase the quality of the metrics that were defined, [22] introduces a set of properties complexity metrics should fulfill. However, it can be observed that the availability of design assessment metrics for object-oriented systems (eg. [5]) was hardly extended over the "object-oriented decade" although some of these metrics have been identified to be also applicable in a service-oriented setting.
The concept of maintainability as defined by the ISO standard 9126 in [10] is narrowed in [1] to the concept of "modifiability". This is achieved by excluding bug-fixing and related activities. Modifiability of a software system is "[...] the ease with which it can be modified to changes in the environment, requirements or functional specification" [1, p. 2]. The concept of modifiability is crucial to the findings of this paper.
The authors of [19] approach an evaluation model for service-oriented systems. It is argued, that quantitative metrics for the maintainability of service-oriented systems would be required. However, they do not provide such

quantitative tools yet.
In [20], for instance, it is argued that coupling and cohesion are meaningful concepts in service-oriented systems. As the article merely motivates to apply object-oriented metrics to SOA, no special metrics for service-oriented systems are defined.
An example of the definition and application of specialized metrics in the area of service-oriented computing can be found in [17]. Types of granularity for single services are introduced as well as metrics for measuring them. However, these metrics are focused on the analysis of single services and not complete systems. In contrast, [21] proposes a system-wide metrics. It applies the concept of coupling to complete *component*-oriented systems.
While the application of metrics to service-oriented systems is sufficiently motivated (eg. by [17], [19] and [20]), the lack of accepted special metrics might be due to both, the little information provided by their application and the complex way to measure them. A set of objective metrics can support organizations approach SO, though.
This paper builds on-top of the work that was done for object-oriented systems and applies it to SO in order to allow for an objective design approach. This is achieved by solely considering properties of services that are available from an outside-view on services and proposing a multitude of related metrics with descriptions of their interpretation. Additionally, a new class of metrics is proposed that address the control-centralization concept of SO. While the concept is largely described (eg., by [7] and [12]), the presented metrics are the first approach toward an objective description of this principle that is known to the authors.
All of those metrics are complementary to any of the approaches that aim at calculating business values, such as the return-on-investment, for service-oriented systems (eg. [13]).

3 Increasing Modifiability with Service Orientation

According to [11], maintainability is the most important quality characteristic when it comes to component-based software development. Being an evolution of the component-based approach (cf. [4]) that adds "soft" design principles (cf. [6]), it can be argued that SO is suitable for increasing a system's modifiability (that is a subset of maintainability). According to [2] and [14], increasing modifiability means to address and reduce complexity.
Another concept for improving the modifiability of a software system is to define an explicit, centralized control model (cf. [7]). One way to incorporate this idea into composite applications is the concept of Business Process Integration Oriented Application Integration (BPIOAI) introduced by Linthicum in [12]. This concept centralizes the control model outside the participating application systems and uses business processes as the central control instance over distributed functionality. This functionality can be exposed by the means of services that have a formally described interface (cf. [12]).
Papazouglou [16] described that SOA allows for business process-centered control over distributed services by introducing process-centered service aggregation – so-called service orchestration. The latter is introduced as a part of a service-oriented architecture. It is a mechanism for aggregating basic services to more specialized services (cf. [16]).
Thanks to the proposed aggregation of services, another possible benefit for the application of SOA within an industry context can be identified: Required changes for functional enhancement could be realized as additional services that are aggregated together with services that expose standard functionality of commercial-off-the-shelf-software (COTS). Such aggregators could provide the required functionality which is offered by separate systems. This way SOA could also contribute to keeping COTS unmodified – which is a major aspect of today's IT governance.
As a quality attribute, modifiability is a subjective, non-functional attribute of composite applications. In order to give organizations and projects a more objectified approach to the modifiability assessment of composite applications, the ideas of service aggregation and control centralization are the basis for the definition of the modifiability metrics that are presented in the next section.

4 Metrics

In this section the necessary basics for the definition and analysis for service-oriented system design metrics are introduced first. Besides the notion of basic definitions, so-called *desiderata* are introduced. The presented properties of size and coupling metrics are cited from prior work that was conducted in the nineties for object-oriented systems.
Based on these basics five new metrics are introduced and discussed. Additionally one metric for component-oriented systems is cited and included in the presented canon of metrics. In total, two metrics for the assessment of a system's complexity and four metrics for measuring the addressing of complexity are discussed.

4.1 Desiderata for Complexity Metrics

This subsection cites properties that size and coupling metrics should satisfy. Additionally, the formal notation that is used for the analysis of these properties is introduced.
In [3], Briand and Basili define a *modular* system Ω as a triple $\Omega =< E, \Omega.R, M >$. E denotes the set of all elements of the system. $\Omega.\Psi \subseteq E$ is the set of all service types in a given system Ω. $\Omega.R \subseteq \Omega.\Psi \times \Omega.\Psi$ denotes the calling

relations among these services. M is an arbitrary collection of disjoint modules of Ω that include all of a system's elements. $OuterR(m)$ denotes the inter-module relations a given module $m \in M$ is involved in. (cf. [3, p. 70]).

Desiderata for *Size* Metrics Based on this definition, [3, p. 71] describes six properties a size metric $\chi(\Omega)$ for a system Ω should satisfy (*desiderata*):

`Size.I` demands that a size metric should satisfy *non-negativity*. It demands that a size metric is not negative.

`Size.II` is the *null value property*: $\Omega = \emptyset \Rightarrow \chi(\Omega) = 0$.

`Size.III` is the *module additivity property* that describes that the size of a system should be equal to the sum of the size of all its modules.

`Size.IV` demands that the size of a system Ω can be determined by the knowledge of the size of its disjoint parts. `Size.IV` is a consequence of `Size.III`.

`Size.V` is the *monotonicity property*. It demands that the value of a size metric must not be decreased if an additional module is added. Monotonicity follows from the properties `Size.I - Size.III`.

`Size.VI` "From the above properties, `Size.I - Size.III`, it follows that the size of a system $[\Omega =< E, \Omega.R, M >]$ is not greater than the sum of the size of any pair of its modules. [...]" [3, p. 71].

Desiderata for *Coupling* Metrics The same authors use the definitions given so far, to propose five properties (desiderata) to which a coupling metric for a modular system should comply to [3, pp. 78–79].

`Coupling.I` is *non-negativity* that demands that a coupling metric shall be zero or greater.

`Coupling.II` demands that the coupling for a system without connections among its modules shall be zero.

`Coupling.III` is the *monotonicity property*. It describes that a new relation between modules does not decrease the coupling

`Coupling.IV` demands that a merger of arbitrary modules should decrease or not effect the coupling value.

`Coupling.V` is a property that describes the merging of unrelated modules in a system. It describes a (modular) system obtained by merging two non-interacting modules as being as complex as the initial system. For the context of service-oriented systems this is interpreted as two services with disjoint methods that are merged together into one bigger service.

4.2 Basic Measures

The basic measures that are introduced here are used in the more complex metrics for complexity and complexity handling measurement. Each measurement attribute is identified with its value range as well as the means to capture it. A discussion part relates it to its impact and later usage.

Number of Services (NS) $NS(\Omega) = |\Omega.\Psi|$.
Mechanism NS is a simple count of all services in a system. It is a *size* metric that only considers the pure count of services in a system.
Value range NS is limited to the range of $[0, +\infty[$
Discussion NS is a simple first measure of a system's complexity: The more services are meant to be used in a system, the more complex (and less modifiable) the system might be. In its simplicity, NS is a basic measure that can be used within other measures and metrics – for complexity metrics and metrics for other characteristics. It is also suitable to weight values of more complex metrics as it provides the context in which these metrics should be analyzed.
NS satisfies the properties `Size.I - Size.VI`

Service Consumers (SC) $\Omega.C$ is the set of all service types that consume (operations of) service providers in a system Ω. $SC(\Omega) = |\Omega.C|$; $\Omega.C \subseteq \Omega.\Psi$.
Mechanism SC is a simple count of all service consumers in a system.
Value range SC is limited to the range of $[0, +\infty[$
Discussion As NS, SC is a basic measure that is used in more complex metrics and denotes the number of service consumers in a system. It satisfies the properties `Size.I - Size.VI`.

Service Providers (SP) $\Omega.P$ is the set of all service types in a system Ω that expose operations that are consumed by service consumers. $SP(\Omega) = |\Omega.P|$; $\Omega.P \subset \Omega.\Psi$
Mechanism SP is a simple count of all service providers in a system.
Value range SP is limited to the range of $[0, +\infty[$
Discussion As NS, SP is a basic measure that is used in more complex metrics. It satisfies the properties `Size.I - Size.VI`.

Service Aggregators (SA) $\Omega.A$ is the set of all service types in a system Ω that both act as service provider and service consumer. $SA(\Omega) = |\Omega.A|$ with $\Omega.A \subseteq \Omega.C, \Omega.A \subseteq \Omega.P, \Omega.A = \Omega.C \cap \Omega.P$.
Mechanism SA is a simple count of all service aggregators in a system. Service aggregators are – as defined by the service-oriented architectural style – sub-types of both service providers and service consumers.
Value range SA is limited to the range of $[0, +\infty[$
Discussion As NS, SA is a basic measure that is used in more complex metrics. The fact that the sets of consumers

and providers overlap in the set of all aggregators is an interesting mechanism that is used in some more complex metrics. The SA value is the most interesting value of the basic metrics as it slightly indicates how a systems complexity is made up and addressed.
SA satisfies the properties `Size.I - Size.VI`.

Coupling of Service (cos) Two components are "coupled if and only if at least one of them acts upon the other" [5, p. 4]. As a service-oriented principle, services should be "loosely coupled" (cf. [6]). Services solely expose operations for interaction with other services. Hence, the *Coupling Between Object Classes* metric (CBO) as defined in [5] is applicable for services as well. This metric is defined as "a count of the number of other classes to which it is coupled" [5, p. 11]. Transferred to services that means that cos is defined as *the count of services a given service calls operations on*. cos is a function of a given service. We denote the cos of a service $s \in \Psi$ as:

$$\cos(s) = |\{\Omega.s\} \times \Omega.\Psi|$$

Mechanism Being a simple count (and not a coupling metric), $\cos(s)$ is a basic indicator for the complexity of a single service. As services encapsulate their variables, in order to calculate $\cos(s)$ it is only checked whether a service uses operations of another service. If so, the value is increased by one. How many operations a service is actually using is not considered.
Value range $[0, +\infty[$
Discussion Still treating a service as a black-box, calculating $\cos(s)$ requires some sort of insight into the mechanisms of a given service. This will be possible to analyze as this needs to be documented for COTS-based services, too. An absolute high value will indicate that the given service depends on many other services. The impact on modifiability depends on the actual class of service that is analyzed. High values for (sole) service consumers might indicate a low modifiability while high values for aggregators might indicate the opposite. Note that $cos(p) = 0$ holds true for a (sole) service provider p.

Inter-Service Coupling (λ) Let $p.\Pi$ be the set of all *receiveCall*-ports of a service provider p. The function π shall be the count of *receiveCall*-ports of a service provider: $\pi(p) = |p.\Pi|$.[1]
Let $c.\Gamma$ be the set of all *serviceCall*-ports of a service consumer. The function γ shall be the count of *serviceCall*-ports of a service consumer c: $\gamma(c) = |c.\Gamma|$.[2]
Let $\Omega.\Lambda$ be the set of all channels between *receiveCall*-ports and *serviceCall*-ports in a system Ω: $\Omega.\Lambda \subseteq c.\Gamma \times p.\Pi$. The function λ is then defined as the cardinality of $\Omega.\Lambda$:

$$\lambda(c, p) = |c.\Gamma \times p.\Pi|$$

Mechanism $\lambda(c, p)$ is the count of channels between the two services c and p. $c \in \Omega.C$ and $p \in \Omega.P$.
Value range The value range of λ is $[0, +\infty[$
Discussion λ is equivalent to the CBO metric as defined in [5]. Therefore it satisfies the same properties – including monotonicity.

4.3 Coupling as a Measure for Complexity

According to [21], the coupling of a system is an indicator for its complexity. This becomes also visible by analyzing the metric properties of [3]. There, coupling and complexity metrics only differ in terms of the *symmetry* property that is defined for complexity but not for coupling metrics. The concept of symmetry addresses the conventions that are used to describe a system.

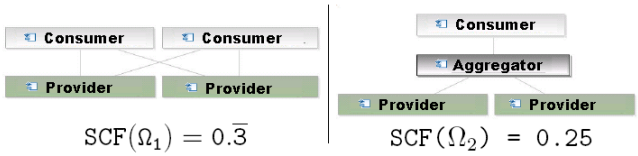
Service Coupling Factor (SCF) In [21], Washizaki et al. have defined a complexity metric called Component Coupling Factor ($CCOF$). As discussed in the introduction, component-orientation and SO are similar architectural styles. Thus, the $CCOF$ complexity measure can also indicate the complexity of a service-oriented system. Using the notation used in this chapter, we define the Service Coupling Factor (SCF) in complete analogy with $CCOF$ as defined in [21]:

$$SCF(\Omega) = \frac{\sum_{c \in \Omega.C} \cos(c)}{NS(\Omega)^2 - NS(\Omega)} \parallel NS(\Omega) \geq 2$$

Mechanism In order to indicate the overall coupling of a given system, the sum over all single cos-values of a system's service consumers is set in relation with the maximum couplings that could occur in a system.
Value range SCF is limited to the range of $[0, 1]$. As by the definition of the service-oriented style, a service consumer needs to be coupled with at least one service provider, 0 will never be seen for service-oriented systems.
Discussion SCF is a metric designed for component-oriented systems that we apply to service-oriented systems, too. Consequently, this metric cannot incorporate service-oriented principles.
As discussed in [21], $CCOF$ (and therefore SCF, too) satisfies the properties `coupling.I - coupling.IV`. Hence, it satisfies the "merging of modules" and "monotonicity". This is because it does not include the notion of service aggregators.
Relatively low SCF values imply a loosely coupled system while high values indicate a dense coupling in a system. SCF can indicate how much the complexity of a system

[1] As a size measure π satisfies the properties `Size.I-Size.VI`
[2] As a size measure γ satisfies the properties `Size.I-Size.VI`

influences its modifiability.
Figure 1 shows some topologies and the corresponding SCF-values.



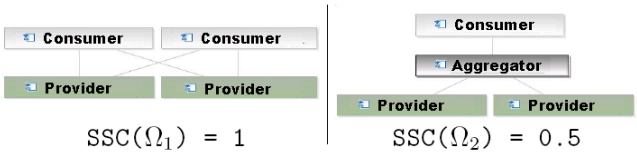
$\texttt{SCF}(\Omega_1) = 0.\overline{3}$ $\texttt{SCF}(\Omega_2)$ = 0.25

Figure 1. Examples of SCF Values

System's Service Coupling (SSC) SSC measures the degree of coupling in a given system Ω with regards to its modifiability. It is defined as:

$$SSC(\Omega) = \frac{\sum_{c \in \Omega.C} \cos(c)}{SC(\Omega) \times SP(\Omega)} \parallel SC(\Omega), SP(\Omega) \geq 1$$

Mechanism In order to indicate the overall coupling of a given system, the sum over all single cos-values of a system's service consumers is set in relation with the maximum couplings that could occur in a system if no aggregators were used at all.
If service aggregators occur in a system Ω, they increase both $SP(\Omega)$ and $SC(\Omega)$. This mechanism increases the denominator and therefore decreases the value of SSC whenever service aggregators are deployed in a system. This is because aggregators are considered to help decrease the overall coupling of services in a system.
Value range SSC is limited to the range of $[0, 1]$. As a service consumer needs to be coupled with at least one service provider, 0 will never be seen for service-oriented systems, though.
Discussion The SSC value indicates to which extent services of a system are cross-linked. The fact that – by definition of the service-oriented architectural style – services need call each other, a SSC value of 0 is not reasonable. If a system gets a relatively high SSC value this is an indication that a lot of interaction without mediation between services takes place. As aggregators automatically decrease the SSC value, a value close to 1 will indicate that a system is hard to modify as it is very complex and not mediated. If a medium value is reached, other indicators should be considered in order to asses the modifiability. This is because systems with a certain level of functionality will need a certain level of coupling, too. Low SSC values indicate a loosely coupled system. Such systems are considered to be better modifiable than more coupled systems.
Whenever a system possesses a high coupling in terms of the SCF value while the SSC value indicates a low coupling, the system's designer obviously tries to address the high coupling by service-oriented principles. However, whether or not this is beneficial is not indicated by these two measures. In such cases other metrics that assess the quality of aggregation should also be considered.
Figure 2 shows some topologies and the corresponding SSC-values.



$\texttt{SSC}(\Omega_1)$ = 1 $\texttt{SSC}(\Omega_2)$ = 0.5

Figure 2. Examples of SSC Values

In order to analyze the presented coupling metric SSC, we consider methods of services as the element of a system, relations among these methods as calling relations (irrespective of any communication semantics) and services as modules that group together sets of methods. This definition fulfills the given definition of element, relation, module and system.
SSC satisfies the following desiderata for coupling metrics:
`Coupling.I`: As neither $\cos(m)$ nor $SC(\Omega)$ nor $SP(\Omega)$ can be negative, SSC obviously satisfies `Coupling.I`.
`Coupling.II`: As $(\forall c \in \Omega \mid cos(c) = 0) \Rightarrow (SSC(\Omega) = 0)$, SSC satisfies `Coupling.II`.
`Coupling.III`: Measuring coupling as described by the SSC metric, $R - OuterR(m) = \emptyset$ holds true. This is because internal calling relations of a single service are not considered. As cos simply counts method calls and additionally simplifies it in a way that all methods a service s calls at another service p can only contribute to the cos-value by 1, the demand that $OuterR(MS') \subseteq OuterR(MS'')$ has the effect that $\sum_{m' \in MS'} \cos(m') \leq \sum_{m'' \in MS''} \cos(m'')$.[3]
Another case to consider is the fact that an additional relation can turn a service consumer (or provider) into a service aggregator. The new relation would always increase the cos value for one service, hence increase $\sum_{m \in MS} \cos(m)$ by 1. On the other hand the new service aggregator would increase $SC(\Omega) \times SP(\Omega)$ by $SC(\Omega)$ (or $SC(\Omega)$, depending on what service is changes). As $SC(\Omega) \geq 1$ and $SP(\Omega) \geq 1$, SSC could be decreased if a channel turns a provider into an aggregator (or vice versa):
$a = \sum_{c \in \Omega.C} \cos(c)$, $b = SC(\Omega) \times SP(\Omega)$
$\frac{a}{b} \geq \frac{a+1}{b+SC(\Omega)}$; $ab + a \times SC(\Omega) \geq ab + b$;$SC(\Omega) \geq \frac{b}{a}$
This means, as soon as more than $\frac{SC(\Omega) \times SP(\Omega)}{\sum_c \cos(c)}$ service consumers are deployed in a system, turning a provider into an aggregator does not increase the complexity of a system. Hence, there are circumstances under which SSC does **not** satisfy `Coupling.III`. As the aim of the metric is to introduce the idea of service aggregators as coupling-reducing mechanism, monotonicity can not be satisfied by SSC.
`Coupling.V` SSC does not satisfy `Coupling.V`. This is because the merger of two (unrelated) services could even

[3]The values might be equal if another relation between two already coupled services is added.

increase the SSC value for a system (what is also a violation of `Coupling.IV` that demands that a merger of arbitrary modules should decrease or not effect the coupling value (cf. [3, p. 78f.])):
By not superseding channels with the merge of services it is possible that $\sum_{m' \in MS'} \cos(m') = \sum_{m'' \in MS''} \cos(m'')$ while $SC(\Omega) \times SP(\Omega)$ is decreased. In these cases the SSC-value is increased (as $\sum_{m' \in MS'} \cos(m') \leq SC(MS') \times SP(MS')$). Thus, merging services only improves (decreases) the SSC value of a system, whenever (bi-directional) relations between services can be spared, too. There also exists a trade-off, in that transforming aggregators into sole providers is considered negative since it results in higher SSC values.
If SSC would be the only rule by which a system is designed, the design would lead to a decentralized and distributed system with numerous services that do not interact heavily and are mediated by aggregators.
SSC violates (under certain conditions) `Coupling.III - Coupling.V`. As SSC incorporates the concept of aggregators as a mechanism to decrease complexity and increase modifiability, the aim of SSC is to reward the use of aggregators. Thus, SSC supports loosely coupled services that are mediated, and are therefore easier to modify than bigger services – even if it does not satisfy all the desiderata for object-oriented coupling metrics.
In order to assess the coupling regardless of service aggregators, SCF should be considered, too. Both metrics can indicate how heavily services of a system interact and whether mediators are used or not. Especially if $SA(\Omega)$ results in a relatively high value, other metrics (eg. SCF and other complexity *handling* metrics that are introduced in the next section) should also be considered.

4.4 Addressing & Measuring Complexity

Extent of Aggregation (EOA) There are two metrics that indicate the degree of a system's aggregation. The first measure is the *Extent of Aggregation* (EOA):

$$EOA(\Omega) = \sum_{c \in \{\Omega.C \setminus \Omega.A\}} \frac{\sum_{a \in \Omega.A} \lambda(c,a)}{\sum_{p \in \Omega.P} \lambda(c,p)}$$

Mechanism EOA relates the count of channels between non-aggregative consumers and aggregators with the overall count of channels from non-aggregative consumers to arbitrary service providers.
Value range The value range of EOA is $[0,1]$.
Discussion EOA describes the extent of hierarchy in the system: The ratio between the count of channels that are mediated by aggregators and the total count of channels from sole service consumers to all service providers. Low values indicate arbitrary channels among a system's components while relatively high values indicate a high degree of aggregation. Usually, a higher degree of aggregation is better as it indicates that complexity is addressed. The EOA value of a system should be looked at with the knowledge of SSC values as well as the knowledge of the absolute counts of the different service types as these values indicate how much complexity a system needs to deal with. Important to note is that EOA relies on a system heavily using consumers. If an application is solely triggered by a service consumer, the result might not be representative for the complete system.
Figure 3 shows some topologies and the corresponding EOA-values.

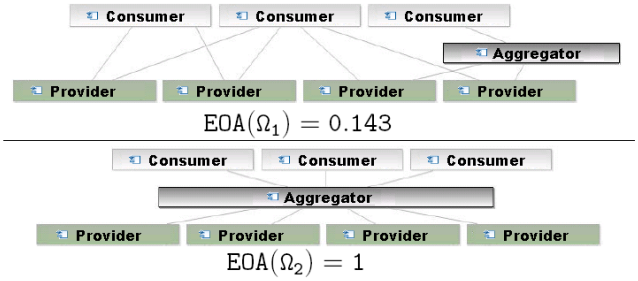


Figure 3. Examples of EOA Values

System's CentraliZation (SCZ) SCZ describes to what extent a system is centralized. Control centralization is seen as a task for service aggregators.

$$\kappa(\Omega) = 0.9 \times (SC(\Omega) - SA(\Omega)) - (SA(\Omega) - 1)^2$$

$$SCZ(\Omega) = 1 - \frac{\sum_{c \in \Omega.C} (\cos(c)) - \sum_{a \in \Omega.A} (\cos(a)) - \kappa(\Omega)}{\sum_{c \in \Omega.C} (\cos(c)) + (SA(\Omega) - 1)^2}$$

Mechanism The basic mechanism of SCZ is to set the extent of a system's consumers' coupling less the coupling of the aggregators in relation with the overall consumers' coupling. The more the aggregators are coupled, the smaller the value gets. Hence, SCZ converges to a value of 1. In order to capture the fact that a service consumer is always coupled with one service provider, the amount of (sole) service consumers is deducted from the denominator. As the use of multiple service consumers is a slight indicator for a de-centralized system, the number of consumers is not deducted completely from the count of service consumers. As the excessive use of (i.e., more than one) service aggregators is contrary to the idea of centralization, a "punishment" for the excessive use of aggregators is also included. This is reflected by the use of the supporting function κ.
Value range The value range of SCZ is $]0,1[$.

Discussion SCZ addresses the need for control centralization in a system that (re-) uses existing parts.
A high value of SCZ indicates that a system uses centralized components. With regards to modifiability, a higher SCZ-value is better than a lower. Important to note is that a high value might be caused by central aggregators but can also be caused by single, central (sole) service providers. This is why the SCZ value for a system can also be high if a system is centralized without control centralization. This can be the case if multiple service consumers use one single service provider. Even if the control is completely decentralized, this case leads to a high SCZ value. This is acceptable since such a hub-and-spoke architecture is also easy to modify (and of course, the system has a central component).
Using multiple centralization components decreases the SCZ value. This is why service aggregators should not be deployed exhaustively. This is especially true for multi-purpose services that act both as consumer and provider without explicit control purposes. This is why the metric includes a "punishment" for the excessive use of aggregators.
A SCZ value close to 1 indicates a high degree of centralization. With regards to an optimal modifiability, such a very high centralization might not be the optimal design for any system, though. This is because a highly complex system that is controlled from one instance might lead to an over-complex, and therefore not modifiable, central control instance. In these cases, a less centralized system might be advantageous. Hence, there is a trade-off between complexity and centralization. This is why the SCZ value of a system should be looked at with the knowledge of SSC and SCF values as well as the knowledge of the absolute counts of the different service types.
Also important to remember is that the metric is based on the assumption that an aggregator centralizes the control of a system. This might not always be the case. Whenever an application uses aggregators in order to adapt to external services, aggregators might be used extensively while the control is centralized in one component. In such cases, the SCZ value can be misleading.
Figure 4 shows some topologies and the corresponding SCZ-values.

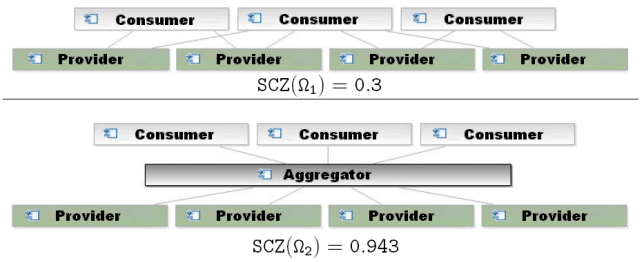


Figure 4. Examples of SCZ Values

Density of Aggregation (DOA) Besides EOA, DOA is the second indicator of a system's degree of aggregation. It indicates *to which extent* the aggregation in a system combines more basic services to more complex services:

$$DOA(\Omega) = \sum_{a \in \Omega.A} \ln\left(\frac{\gamma(a)}{\pi(a)+\gamma(a)} \times 2\right)$$

Mechanism DOA relates for each service aggregator the count of *serviceCall*-ports to the overall count of the service's ports. The value range for this ratio is $]0, 1]$. A value of ≥ 0.5 for an aggregator indicates that it consumes more ports than it provides. By multiplying it by 2 and calculating the logarithmic value for this result, such aggregators get a low positive score. For aggregators that offer more ports than they consume, a relatively high negative value is the result.
Value range The value range of DOA is $]-\infty, +\infty[$.
Discussion By "punishing" the non-aggregative use of aggregators with relatively high negative values while "rewarding" only low positive scores to "real" aggregators, an overall positive value indicates proper use of aggregators. The absolute value of this measure is irrelevant. Especially in combination with the SCZ the DOA value can indicate whether centralization of a system goes along with a good aggregation. Hence, for high SCZ values, the DOA value should be positive. If a negative DOA meets a high SCZ value, a system might use improper centralization mechanisms that decrease the level of modifiability.
Figure 5 shows some topologies and the corresponding DOA-values.

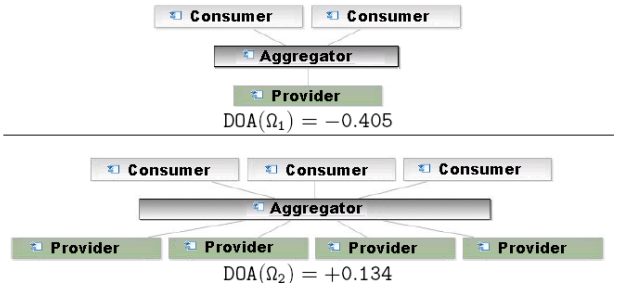


Figure 5. Examples of DOA Values

Aggregator CentraliZation (ACZ) ACZ indicates the degree of centralization in a system by considering the use of mediating services. Mediators are identified by the supporting measure AD.

$$AD(\Omega, a) = \begin{cases} 0 & \text{if } 0.5 \leq \dfrac{\gamma(a)}{\pi(a)+\gamma(a)} \leq 0.\overline{6} \\ 1 & \text{otherwise} \end{cases} \parallel a \in \Omega.A$$

$$ACZ(\Omega) = \begin{cases} 1 & \text{if } SA(\Omega) = 1 \\ 1 - \dfrac{\sum_{a \in \Omega.A} AD(\Omega, a)}{SA(\Omega)} & \text{otherwise} \end{cases}$$

Mechanism ACZ combines the idea of control centralization via aggregators and considers the actual density of an aggregation. An aggregator that does not compose several services is indicated by an *Aggregator Density* (AD) of 0. AD incorporates the idea that no to little control is executed if the density of an aggregation is low. Such an aggregator is considered to be a *mediator*. If services that consume one service and also act as service providers should be seen as service mediators, the AD measure needs to adjusted accordingly ($\frac{\gamma(a)}{\pi(a)+\gamma(a)} = 0.5$).
By relating the count of non-mediators with the overall count of service aggregators in a system the ratio of such aggregators is calculated. By deducting this ratio from 1 the degree of centralization into non-mediators is indicated.
Value range The value range of ACZ is $[0, 1]$.
Discussion ACZ can be used as a measure to interpret SCZ values. SCZ describes to which degree a system mediates service calls and interprets the use of few aggregators as a centralization. Without considering the internals of an actual aggregator, this can be a misleading interpretation. As the analysis of component internals is considered to be hardly applicable in real-life settings, the interpretation of the internals is supported by the ACZ metric. The ACZ metric incorporates the assumption that control can only be exercised by a service aggregator whose count of *receiveCall*-ports is disparate from the count of *serviceCall*-ports. Values close to 1 indicate a high degree of centralization while values close to 0 indicate a low degree of centralization in a system. A complete centralization in terms of a ACZ-value of 1 can only occur if only one aggregator is used or only mediators are used. In real-life settings this unlikely to occur.
ACZ is valuable for the interpretation of the results of the SCZ metric for a given system: If a system's design scores a relatively low SCZ-value, a high ACZ-value indicates that the mediators are used in the design of the system and that the design incorporates the idea of control centralizations. However, if both the ACZ and the SCZ values are low, the system does not follow a centralized control model. Of course, this metric has also to be carefully applied. This is because it also solely puts an interpretation of an externally visible structure over the actual internal structure of a component that determines the visible part. In conjunction with the SCZ, DOA and EOA values it is considered valuable, though.
Figure 6 shows some topologies and the corresponding ACZ-values.

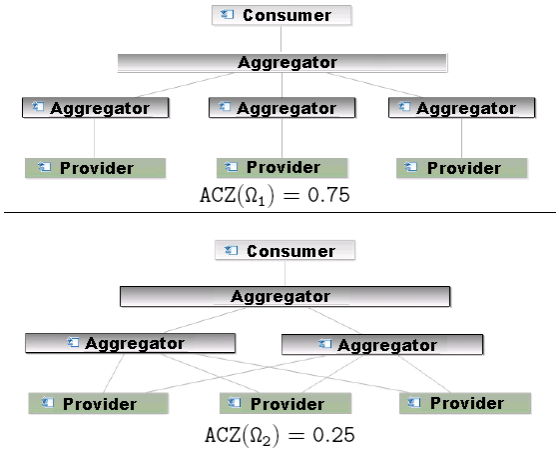


Figure 6. Examples of ACZ Values

5 Interpreting the Metrics

All metrics that were described in the previous section indicate the modifiability of a system – some as indicators for high modifiability, some as indicators of low modifiability. An optimal mechanism would be to identify a boolean discriminant function (BDF) with according thresholds for the single metrics in order to determine the modifiability of a system that realizes the analyzed design (as described similar in [18]). This approach has two problems. Realizing a significant application that goes beyond scientific prototyping is cost intensive. This is why the first issue is to get relevant data for running regression tests. A second issue is the required categorization of systems into "easy to modify" and "hardly modifiable". We do not consider such a fragmentation feasible, as it will heavily reflect subjective parameters. This is why an aggregated discrimination function that forecasts the modifiability of a system is considered unrealistic.

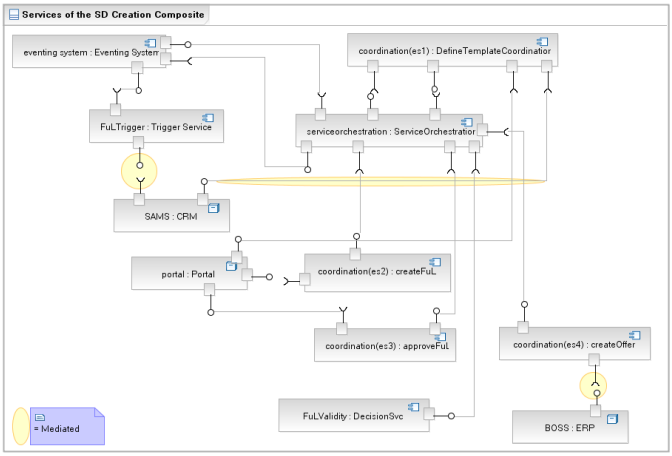


Figure 7. Services and Relations

We believe that the introduced metrics can be better used to highlight certain aspects for service-oriented design principles and their impact on modifiability. This is why a qualitative description of inter-relations among the metrics is considered better applicable than a quantitative discriminant function. As a rule it can be stated that if the metrics give a "bad" picture of a design, a redesign is appropriate. If the picture is "good", the design can still be suboptimal.

Application (Ω)	NS	SC	SP	SA	SSC	SCF
SD Creation	11	8	11	8	**0.156**	**0.109**

Table 1. Complexity Metrics for the Example Composite Application

They are a necessary indicator but not sufficient indicators of design quality. For the sake of demonstrating their applicability, the following case study shows the metrics in the context of real-life requirements.

5.1 A Case Study

The case study to which the above metrics are applied is a composite application that was designed (and realized) as part of a project in the context of an industry company. The design that was made based on the to-be implemented business process was aligned with a reference architecture for composite applications (cf. [9]). The design of the application is described in [8].

The business process that is (semi-) automated by the composite application describes the procedure how the company reacts on customer demands by estimating the efforts the realization of a request might cause and providing an offer to the customer. In the overall process there exists an extract that is concerned with creating an offer for a given demand. The "service description" (SD) is a document that is used to describe a solution that is offered in response to a demand. An SD accompanies an offer and delineates the offered services, contains (among others) a functional description of the solution and indicates the service-level of the offered services. This description needs to be aligned with the service portfolio the company offers. Only services that are officially included in the company's portfolio can be proposed. A SD is the base for a cost calculation for the respective solution. Based on the estimated cost, prices are made and an offer, that includes a SD as well as the calculated prices, is sent to the customer.

A rough overview of the composite application's design is summarized in the component diagram in figure 7. In this diagram, services are modeled as components. Service consumers have *required* interfaces while service providers offer *provided* interfaces. Service aggregators have both types of interfaces.

In order to describe the complexity of the composite application for the SD creation process, the complexity metrics are applied to the composite application. The results of this analysis are shown in table 1.

The analysis shows that the complexity metrics SSC and SCF produce low values. This means that the overall complexity (by the notion of coupling) of the composite application can be assumed to be relatively low. These values solely indicate the coupling of the services. To get another picture of the composite's complexity, the count of the single services should be considered, too.

In order to provide an overview of the behavior of the complexity handling metrics, they are also described for the SD creation composite.

Interesting to note is that the extent of aggregation is zero. This is because no pure service consumers are part of the system. This is also why the system's centralization, as far as the SCZ is concerned, is relatively low.

Another reason for a relative low centralization value is that the application services are mediated by service coordinations with a low density. As the service coordinations do not provide any control logic, the SCZ, that is only based on the notion of aggregators, indicates a distributed control. As discussed, should the SCZ value be considered in combination with the SSC and SCF values. As these values indicate a low complexity, the little centralization in terms of the SCZ metric is acceptable.

The ACZ metric indicates a relative high degree of centralization. This is because most of the aggregators are used as mediators.

The density of aggregation (DOA) is positive. This means that the aggregators access more service methods than what they provide. Hence, the aggregation is *sufficiently* dense. The values of the complexity handling metrics are shown in table 2.

With regards to the metrics that describe the modifiability of a system it can be summarized, that the overall picture for the SD creation composite indicates an application with overall low complexity that extensively uses (appropriate) aggregators. On the one hand, the use of aggregators is a sign of incorporating service-oriented principles. On the other hand, the extensive use of aggregators endangers the principle of control centralization in terms of the SCZ metric. The reason for the extensive use of aggregators is the adaption of heterogeneous applications to a functional description of a business process. The fact that aggregators are used as mediators is objectified by the relatively high ACZ value for the scenario. This value indicates a high control centralization within the aggregators. Expressed differently, 75% of the aggregators is used as mediators while the control is centralized in 25% of the composite application's aggregators. Seen from this point of view, the SD creation composite centralizes the control on top of a heterogeneous landscape. The overall picture that is provided by this analysis is that the composite applications will show a high degree of modifiability if future requirements need to be reflected. This is why the metrics do not indicate the need for a redesign.

Application (Ω)	SCZ	EOA	DOA	ACZ
SD Creation	**0.22**	**0**	**+0.4**	**0.75**

Table 2. Complexity Handling Metrics for the Example Composite Application

6 Conclusion and Future Work

In this paper we have presented a set of metrics that can support large organizations to objectively incorporate some simple principles of SO into their application design. By using this tool, they can more easily leverage service-oriented principles that are *soft* in comparison with *hard* principles that are equivalent to the component-oriented architectural style. These metrics are a contribution to the application of the service-oriented architectural style in the context of large organizations. They solely provide a necessary indication of design quality. They are not sufficient indicators, though.

The metrics presented here are an initial proposal that proved to be beneficial during an application in a first larger real-life project. Especially as part of a design methodology for composite applications, the metrics can help to identify a poor design and motivate a re-engineering of a design prior to realizing the application. However, the definition of the metrics can only be a first step toward the objective application of the service-oriented architectural style. As discussed in this paper, a boolean discriminant function with fixed thresholds would be preferable but is considered to be not achievable today.

It is required to apply the metrics in additional settings to get a more objective base for their interpretation. This is why we will continue to research the behavior and significance of the presented metrics in more enterprise-scale case studies.

References

[1] P. Bengtsson, N. H. Lassing, J. Bosch, and H. van Vliet. Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software*, 69(1-2):129–147, 2004.

[2] L. Bratthall and P. Runeson. A Taxonomy of Orthogonal Properties of Software Architecture. *Proc. 2nd Nordic Software Architecture Workshop. Ronneby, Aug*, 1999.

[3] L. C. Briand, S. Morasca, and V. R. Basili. Property-based software engineering measurement. *IEEE Trans. Softw. Eng.*, 22(1):68–86, 1996.

[4] H. Cervantes, L. Imag, and F. Hall. Technical Concepts of Service Orientation. *Service-Oriented Software System Engineering: Challenges and Practices. Idea Group Publishing*, 47, 2005.

[5] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Software Eng.*, 20(6):476–493, 1994.

[6] T. Erl. *Servcie-Oriented Architecture*, volume Fourth Printing of *The Prentice Hall service-oriented computing series*. Prentice Hall, Inc., Upper Saddle River, NJ USA, February 2006.

[7] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In *ICSE*, pages 179–185, 1995.

[8] H. Hofmeister and G. Wirtz. Designing a platform-independent use-case for a composite application using a reference architecture. In *Proceedings of the 19th Int. Conf. on SW Eng. & Knwl. Eng. (SEKE'2007), Boston, MA, USA, July 9-11, 2007*, 2007.

[9] H. Hofmeister and G. Wirtz. A multi-layered framework for pattern-aided composite application design. In *The 11th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI'2007), Orlando, FL; USA*, volume 1, pages 54–60, 2007.

[10] ISO/IEC. Information technology – software product evaluation. Technical report, International Organization of Standardisation and International Electrotechnical Commission, 2004.

[11] K. Lee and S. J. Lee. A quantitative evaluation model using the iso/iec 9126 quality model in the component based development process. In *ICCSA (4), volume 3983 of Lecture Notes in Computer Science*, pages 917–926, 2006.

[12] D. S. Linthicum. *Next Generation Application Integration*. Addison-Wesley, Boston, MA USA, 2004.

[13] E. A. Marks and M. Bell. *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2006.

[14] T. J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.

[15] A. Nori and R. Jain. Composite applications: Process based application development. In *TES*, volume 2444 of *LNCS*, pages 48–53. Springer, 2002.

[16] M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *WISE*, pages 3–12. IEEE Computer Society, 2003.

[17] D. Rud, S. Mencke, A. Schmietendorf, and R. Dumke. Granularitätsmetriken für serviceorientierte Architekturen. In *DASMA Software Metrik Kongress (METRIKON'2007)*, 2007.

[18] N. F. Schneidewind. Software metrics model for quality control. In *IEEE METRICS*, pages 127–136. IEEE Computer Society, 1997.

[19] M. Stutz and S. Aier. Vorgehensmodell zur fachlichen Bewertung serviceorientierter Architekturen. In *Multikonferenz Wirtschaftsinformatik*. GITO-Verlag, Berlin, 2008.

[20] S. Vinoski. Old measures for new services. *IEEE Internet Computing*, 9(6):72–74, 2005.

[21] H. Washizaki, T. Nakagawa, Y. Saito, and Y. Fukazawa. A coupling-based complexity metric for remote component-based software systems toward maintainability estimation. In *APSEC '06 Proceedings*, pages 79–86, Washington, DC, USA, 2006. IEEE Computer Society.

[22] E. J. Weyuker. Evaluating software complexity measures. *IEEE Trans. Softw. Eng.*, 14(9):1357–1365, 1988.