On the State of the Art of Coupling and Cohesion Measures for Service-Oriented System Design

Dominik Q. Birkmeier
University of Augsburg, dominik.birkmeier@wiwi.uni-augsburg.de

Follow this and additional works at: http://aisel.aisnet.org/amcis2010

Recommended Citation
http://aisel.aisnet.org/amcis2010/296

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
On the State of the Art of Coupling and Cohesion Measures for Service-Oriented System Design

Dominik Q. Birkmeier  
University of Augsburg  
dominik.birkmeier@wiwi.uni-augsburg.de

ABSTRACT

Service-oriented computing has encountered an increasing importance for enterprises over the last years. With Web services, the major underlying technical basis is already in an advanced state. The service design area, on the other hand, still provides several research gaps such as the field of service identification and in particular the determination of an optimal granularity level for services. Granularity, assessed through coupling and cohesion considerations, is yet a rather unexplored domain when it comes to service-orientation, although several results from earlier design principles are available. In this paper we summarize the current state of the art in granularity measures and identify the implications emerging for practice and research. As we reveal, several existing measures for other paradigms, which might be adapted for service-orientation, are left unconsidered. Further research gaps, as the mainly missing empirical evaluation or a tighter inclusion in the development process, are also detected.

Keywords

Service-oriented enterprise, service-oriented design, coupling, cohesion, measures.

MOTIVATION

Over the last decade, service-oriented computing (SOC) has emerged as a major challenge and opportunity for enterprises. More than ever, enterprises need to adapt their processes to new market developments in a highly agile and flexible way, while maintaining or even increasing their effectiveness and efficiency. The service-orientation paradigm offers enterprises the potential for such a greater organizational agility (Erl, 2005; Papazoglou and Georgakopoulos, 2003). In this course, the SOC concept, has rapidly spread over the last years as a solution for technical problems and application designs, in conjunction with Service Science Management and Engineering (SSME) for business processes (Chesbrough and Spohrer, 2006).

With the expansion of SOC designs, more and more enterprises are concerned with the success of such implementations, which seems to be reasonable since so far virtually no empirical research on the benefits and drawbacks of Service-Oriented Architectures (SOA) has been done (Erickson and Siou, 2008a, b). With Web services (WS) as the current technique to access service-oriented (SO) solutions (Erl, 2005), the major technical middleware such as SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery, and Integration), WSDL (Web Service Description Language) and XML (Extensible Markup Language) is already in a mature state (Ciganek, Haines and Haseman, 2005; Curbera, Khalaf, Mukhi, Tai and Weerawarana, 2003). Consequently, several of the key research gaps are mainly located in the service design area. Vitharana, Bhaskaran, Jain, Wang and Zhao (2007) for example state the challenge to embody domain knowledge and corresponding functionalities in loosely coupled services that could be used to compose applications for the domain and which additionally are separately marketable. These tasks, and the examination of methodologies for aligning business processes with corresponding services, can be accounted to the field of service identification, which is currently being researched in several directions (Birkmeier, Klöckner and Overhage, 2009a). Future research, furthermore, needs to analyze the topic of granularity, which “plays a key role in service design” as too coarsely as well as too finely granular services both have drawbacks (Vitharana et al., 2007). Erl (2005) agrees that “service interface granularity is a key strategic decision point that deserves a good deal of attention during the service-oriented design phase”.

Determining the optimal granularity of design artifacts is not a completely new task coherent to service-orientation. For large parts it can be adapted from earlier paradigms such as component-oriented (CO) programming, object-oriented (OO) programming or, more general, modularization principles. Thus, the aspects of scale and granularity can be ascribed to modularity criteria (Szyperski, Gruntz and Murer, 2002). Common modularity considerations go back to Parnas (1972) and the principle of maximizing cohesion while minimizing dependencies. This principle also forms the basis of several component iden-
tification approaches (Birkmeier and Overhage, 2009b), as well as upcoming service identification approaches. However, as assessing coupling and cohesion of a certain structure is rather not a straightforward task, there exists a variety of different metrics. Furthermore, as explained later on in more detail, there are also several differences between the SO, CO and OO paradigms and, thus, existing measures cannot equally be applied to all of them. Accordingly, comparing existing measures and choosing the right one for a certain application is challenging.

The main purpose of this paper, hence, is to provide an overview on the state of the art in granularity measures for service-oriented computing, and thereupon deduce implications for practice, as well as academia. To determine the eligible measures and the factors upon which they are compared, we conducted a thorough literature survey in journals and conferences in the information systems and software engineering disciplines. Our research approach thereby follows the methodology for literature reviews as illustrated by Webster and Watson (2002). For the remainder of the paper we will proceed as follows: after presenting related work and motivating the research gap in the next section, we will discuss different possibilities to assess service granularity in general. Thereupon, we derive a comprehensive classification schema for existing and future coupling and cohesion measures. Furthermore, we describe and evaluate several common approaches in more detail. Before we sum up and provide an outlook for our future work, we conclude with implications arising for research and practice.

RELATED WORK

There exists a large variety of coupling and cohesion measures. All of them have different purposes and application areas. Nevertheless, the evaluation and comparison of such measurements has only sporadically been addressed in literature, often with a specific focus. Zhou, Lu, Lu and Xu (2004) for example analyze six different graph-theory-based cohesion measures for classes in OO systems. They perform a purely theoretical comparison of them with respect to the types of nodes, the types of relationships, the theoretical validation, the discriminability, and the complexity. Unfortunately they do not provide any detailed recommendations, but conclude that “to measure the cohesiveness objectively, a cohesion measure should have some good properties” (Zhou et al., 2004). In their work on design-level cohesion measures for modules, Kang and Bieman (1996) formally define a new cohesion measure, which does not rely on code, but utilizes an input-output dependence graph and thus can be used ahead of any implementations. They compare the introduced Design-Level Cohesion (DLC) and Design-level Functional Cohesion (DFC) measures to previous Functional Cohesion (FC) measures. However, they do not perform an in-depth analysis of the state of the art.

Briand, Daly and Wüst conducted a detailed examination of various different coupling and cohesion measures in OO systems. They derived a unified framework for cohesion measurement (Briand, Daly and Wüst, 1998), as well as a distinct one for coupling measurement (Briand, Daly and Wüst, 1999a). In both cases they separate between rather subjective, qualitative classifications of different coupling/cohesion levels, and more objective, quantitative measures, which are calculated from different criteria. In the beginning, they define a joined terminology and formalism as a common basis for all approaches. Following the recapitulation of existing measures, they perform an expansive comparison that provides the input for their own, unified frameworks. Overall, the studies are limited to measurements in OO systems and, furthermore, do not cover several of the newer approaches available.

Perepletchikov and his colleagues were the first to perform explicit studies on coupling and cohesion metrics for SO designs (Perepletchikov, Ryan and Frampton, 2007a; Perepletchikov, Ryan, Frampton and Tari, 2007b). In particular, they elaborate on possibilities for predicting the maintainability of SO software in an early design phase through structural attributes in order to be able to judge the software quality as early as possible. Consequently they are focused on rather technical aspects of the systems. To our best knowledge, there has been no thorough comparative analysis on the current state of the art in coupling and cohesion measures for SO systems.

GRANULARITY OF SERVICES

In software development, determining the optimal granularity of an artifact is a common problem, frequently addressed over time. As the discussion of related work revealed, however, it has rarely been examined in a SOC context. Nevertheless, it is of high importance especially for the identification of possible service candidates (Vitharana et al., 2007), as well as for maintainability considerations of SO software (Perepletchikov et al., 2007a; Perepletchikov et al., 2007b). In both cases, granularity can be expressed through coupling and cohesion metrics, which belong to the group of internal (structural) quality attributes (Chidamber and Kemerer, 1994). Generally speaking, coupling and cohesion can be understood as follows (Stevens, Myers and Constantine, 1974):

Coupling is a measure of interdependencies between elements within a module (internal coupling) or between different modules (external coupling).

Cohesion measures how tightly the internal elements of a module are related to one another.
Following the modularity principles of Parnas (1972), high cohesion and low coupling is favorable for a modular system design. Regarding the maintainability of SO software, such a design, furthermore, promises to be easier to analyze and test with an improved stability and changeability (Perepletchikov et al., 2007a). Today, especially the possibility to incorporate quality measures in the early software development phases is of increasing importance for the design determinations.

Despite the fact that measures specifically adopted for the SO paradigm are still rare, most authors agree upon their necessity (Erickson et al., 2008b; Vitharana et al., 2007). On the other hand, Steve Vinoski (Vinoski, 2005) proposes in his comment on “Old Measures for New Services” that for the fundamental concepts of coupling and cohesion, existing measures from structured design and OO design could be reused. However, he solely discusses qualitative measures, rather than quantitative metrics. Furthermore, we agree with Perepletchikov et al. (2007a; 2007b) that existing measures are not immediately applicable to service-oriented computing, due to several distinctive characteristics of the SO paradigm. First of all, SOC more than ever follows the key development principles to provide highly flexible and agile systems with ideally reusable and separately marketable services. Furthermore, compared to procedural programming with one level of abstraction (i.e. procedures) and OO software with two abstraction levels (i.e. methods aggregated into classes), the SO design introduces an additional abstraction layer (i.e. services providing the functionality of their inner elements through service interfaces). The boundary of services, however, is logical rather than physical and encapsulates elements, which can be implemented in various languages. Due to these differences, existing measures are not necessarily applicable to the SO paradigm in a straightforward manner.

The majority of the metrics discussed later on is based on graph-theory. Using a graph-theoretical approach is very common for most ways of posing a problem. In OO systems for example, class elements such as attributes and methods are represented through nodes, and relationships between them refer to edges in the graph (Zhou et al., 2004). This methodology is applicable to SO designs as well, since graphs can be mapped onto the different abstraction layers of SOC. Furthermore, graph-theoretical approaches are common for service identification procedures as well. In bottom-up identification approaches, graphs can depict technical information resulting from code, etc. and for top-down methods design information like business processes, data models, etc. can be used (Albani, Overhage and Birkmeier, 2008; Birkmeier et al., 2009a; Jain, Chalimeda, Ivaturi and Reddy, 2001). Additional information like weights, node-types, etc. can be assigned to the elements of a graph. Thus, graph-based measures can be considered as universal approaches, applicable to several different scenarios, independent of a specific modeling notation. Related disciplines as for example compactness measures in the structural analysis of hypertexts operate in similar ways (Botafogo, Rivlin and Shneiderman, 1992).

Figure 1. A simplified excerpt from an Academic Management System (AMS)

Figure 1 depicts a simplified excerpt from an Academic Management System (AMS). Once it is reduced to the internal elements and the interfaces, together with their relations, it is easy to see how the information can be mapped onto a directed graph. Different measures of coupling and cohesion are attached to different aspects of the graph. One possible way to assess coupling is through an examination of the external communication between interfaces. Cohesion, on the other hand, is usually implemented as a normalized measure, which, for example, utilizes the ratio of actual to possible communication in a service.
Classification Schema

For a thorough comparison and classification of current and future measures, a set of characteristic criteria is described in the following. The resulting schema is equally applicable to coupling and cohesion measures. Besides the unified frameworks of Briand et al. (1998, 1999a) and the criteria used by Zhou et al. (2004) for measures in OO systems, it is further based on the characteristics of SO design described by Perepletchikov, Ryan, Frampton and Schmidt (2008), as well as the authors experience in SO design and measures. In addition to general information like the name, a coarse definition, and the source of the measure, several criteria are collected in three categories. First of all, every metric is based on conceptual foundations, which describe the understanding of the central concepts and the general characteristics. Furthermore, each measure is built upon an underlying model that includes the essential properties, relevant to the measurement procedure. At last, the measurement characteristics describe attributes as for example the level of validation. While the following criteria might not necessarily be complete, they have repeatedly been used to adequately summarize software measures in theory (Briand et al., 1998, 1999a).

Conceptual Foundations

The underlying design paradigm reveals if a measure is specifically developed for service-oriented, component-oriented, object-oriented or procedural system design. It is safe to assume that measures are best applicable in the paradigm they are developed for.

Whether a measure can be determined objectively, or if it is influenced by subjective preferences, is expressed in the objectivity criterion. Qualitative measures, often called frameworks, usually depend on the evaluation of the user and, thus, are difficult to automate and use in practice. Quantitative metrics, on the other hand, are automatically computable and do not necessarily depend on personal preferences (Briand et al., 1999a; Kang et al., 1996).

The domain of the measure divulges whether it is applicable at the design or the code level.

The level of measurement criterion determines the type of scale, the measure is defined on. Possible types of scale are in ascending order: ordinal, nominal, interval or ratio. A higher level of measurement is generally preferable since it enables more detailed operations (Stevens, 1946).

A measure can be derived from different types of measurement. It might be calculated on the basis of a graph-theoretical representation, a design model (e.g. a Unified Modeling Language (UML) model) or the implementation code, or it might be achieved from conceptual considerations. This is independent from the domain of the measure, as for example models or graph-theoretical representations might be automatically generated from code.

Underlying Model

For graph-based and model-based measures, the type of nodes criterion lists the elements of a service, class, business domain, etc., which are mapped onto the nodes of the graph or are part of the model. Measures that do not only incorporate standard elements, as for example internal classes in services, or attributes and methods in OO classes, but also specialized elements such as service interfaces or constructors, are supposed to be more accurate (Zhou et al., 2004).

Similar to the node types, graph- and model-based measures should discriminate different types of connections. Those are within others: references, interactions, sequences, etc. Connections are considered as the “mechanisms that constitute coupling” and “make a class cohesive” (Briand et al., 1998, 1999a). Again, it can be assumed that more accurate measures differentiate more specialized connection types (Zhou et al., 2004).

Measures might additionally account for different strengths of connections, for example through an inclusion of weights. Indirect connections are not respected in all measures, as oftentimes solely the direct connections are counted. For example: if a method m3 follows m2 and m2 follows m1; are, consequently, m1 and m3 related as well? (Briand et al., 1998)

A criterion specific to coupling measures is the direction of coupling. It reveals whether from the systems perspective the measure is restricted to internal and/or external coupling, or if the direction is irrelevant.

Measurement Characteristics

Operationally defined indicates whether a measure in its original definition is immediately applicable for the respective development principle, or if an additional interpretation is necessary (Briand et al., 1998).

There are two aspects of complexity for a measure. The first one arises from the constructing complexity of the abstract model (e.g. mapping onto graph) and the second one is the computing complexity. Whichever is higher determines the overall complexity of the measure (Zhou et al., 2004).

The criteria usable and partially usable address the question when in the development process the measure becomes applicable. It is partially usable, when all information is available at the end of a development phase, but can be further re-
defined in the later process. It is usable, if all necessary information is stable. For this, we assume an underlying sequential development process with the generic phases: analysis (An), high level design (HLD), low level design (LLD) and implementation (Imp) (Briand et al., 1998).

The final criterion is probably the most important one for the applicability of a measure. It expresses the current state in the validation of the measure. In the worst case, no validation is reported and, thus, the measure is hardly applicable for a design critical to the overall system. Ideally a measure is evaluated theoretically and empirically. A theoretical validation should be done against a formal software engineering measurement framework. For example, by addressing Briand’s properties for coupling and cohesion measures: (C1) nonnegativity and normalization, (C2) null value and maximum value, (C3) monotonicity and (C4) merging of unconnected classes (Briand et al., 1998, 1999a). For an empirical validation, the measure has been investigated with respect to its causal relationship on an external quality attribute. Measures are usually only validated according to their original design paradigm.

The derived distinguishing factors are the basis for the classification schema depicted in Table 1. Thereby, values of the identified factors have been summarized as a morphological box and can be used to classify current and future measurement approaches. From looking at the classification schema as a whole, one might suspect that the depicted distinguishing factors are not independent from each other. Analyzing the factors more closely, however, reveals that they are all necessary to characterize measures sufficiently.

<table>
<thead>
<tr>
<th>Classification Schema for Coupling and Cohesion Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design paradigm</td>
</tr>
<tr>
<td>Domain of the measure</td>
</tr>
<tr>
<td>Level of measurement</td>
</tr>
<tr>
<td>Type of measurement</td>
</tr>
<tr>
<td>Development model</td>
</tr>
<tr>
<td>Measure characteristics</td>
</tr>
<tr>
<td>Type of nodes (*)</td>
</tr>
<tr>
<td>Type of connections (*)</td>
</tr>
<tr>
<td>Strength</td>
</tr>
<tr>
<td>Indirect Connections</td>
</tr>
<tr>
<td>Direction of coupling (*)</td>
</tr>
<tr>
<td>Operationally defined</td>
</tr>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>Usable / Partially usable</td>
</tr>
<tr>
<td>Validation</td>
</tr>
</tbody>
</table>

Table 1. Classification schema for coupling and cohesion measures

**Coupling and Cohesion Measures**

Based on the schema developed in the previous section, current coupling and cohesion measures for SOC as well as for other paradigms are classified in the following. The single measures, or measure suites, are briefly introduced in chronological order. In the interest of brevity, the reader is referred to the referenced publications for details on the measures. Table 2 summarizes all classification results for coupling measures and Table 3 for cohesion measures, respectively.

**SO Coupling Measures**

In his earlier discussed paper Vinoski (2005) demands to reuse measures from earlier paradigms for service-orientation. In this context, he also provides an example of adopting the coupling categories from Page-Jones (1988) and Stevens et al. (1974). He discriminates between the normal types of coupling (data coupling, stamp coupling and control coupling), which are generally okay and three types that should be avoided (common coupling, content coupling and interface coupling). However, these qualitative classifications are rather imprecise and strongly depend on the user’s interpretation.

Perepletchikov et al. (2007b) argue on the applicability of common object-oriented and procedural metrics for the SOC paradigm. As they conclude that such metrics are not directly applicable, they introduce a whole suite of new coupling measures, with the focus on predicting maintainability in SO designs. Doing so, they derive eight assumptions on analyzability, changeability, stability and testability, for which the new measures are required. Their nine primary metrics, which can be directly calculated from a graph representation of the system, are later on assembled to eight aggregated metrics. An empirical evaluation of the metrics and the assumptions, however, is left for future work.
Coupling Measures for Other Paradigms

Li and Henry (1993) provided the idea for two popular metrics, which predict maintainability in OO systems. Message Passing Coupling (MPC) utilizes the number of send statements sent from a class, as this indicates how dependent the local methods are on the methods of other classes. Furthermore, they define Data Abstraction Coupling (DAC) as the number of abstract data types defined in a class. For both measures, they left the realization of their ideas to others (e.g. Briand et al. (1999a)).

Another popular set of OO metrics was introduced by Chidamber et al. (1994), who propose the straightforward measure: Coupling Between Objects (CBO) as the count of the number of other objects to which an object is coupled. However, they do leave the decision on what types of connections are really relevant to others. The Response for Class (RFC) metric, furthermore, counts the “methods that can potentially be executed in response to a message received by an object of that class”.

Based on the previous measures, Hitz and Montazeri (1995) aim at providing an improved, graph-theoretical version of them. They introduce new aspects like the stability of client and server classes and the separation between class-level and object-level coupling. Overall, their comprehensive framework for all sorts of coupling, however, remains a rather subjective, qualitative set of coupling types.

An investigation of coupling measures with focus on their applicability for C++ was performed by Briand, Devanbu and Melo (1997). Besides the introduction of a large framework of subjective coupling types, they furthermore proposed a whole suite of coupling measures. Most of them are designed for C++, but also several general metrics are included. They are specifically applicable at different development phases and serve variable purposes. As Briand et al. where the ones who introduced a common set of rules for theoretical validation, all of the measures fulfill these requirements. Additionally, all of them are empirically evaluated and, thus, are frequently referenced.

Allen and Khoshgoftaar (1999) approached coupling measurement from an information-theory perspective. They introduced graph-based measures for intermodule and intramodule coupling, applicable to general modules (e.g. software components). The mapping of a problem onto a graph, however, is not specified, but left to the user. Thus, it is a very general approach, which is, nevertheless, evaluated theoretically against Briand’s properties.

For their strategy-based design of reusable business components, Vitharana, Jain and Zahedi (2004) link managerial goals (e.g. cost effectiveness) with technical features (e.g. coupling). The defined coupling metric measures the extent to which classes within different components are related to each other. This is done by adding up all coupling weights of connections between classes in different components. However, they do not provide any details upon which connections are considered.

A newer approach for component-based systems was recently claimed by Gui and Scott (2007) and compared to traditional coupling metrics. Their focus, hereon, was placed on measures aiming at the evaluation of Java component reusability with the goal of future automatic component identification in the internet. The theoretical and empirical evaluation of the new measures, however, felt rather short. Furthermore, the specialization on components is missing, as they still solely operate on OO classes and methods.

SO Cohesion Measures

As for coupling, Vinoski (2005) proposes several SO types of cohesion. Functional, sequential and communicational cohesion are considered as “good” forms, whereas procedural, temporal, logical and coincidental cohesion are frowned upon. Again, those measures are subjective and imprecise.

Similar to their work on coupling measures, Perepletchikov et al. (2007a) present five basic and one assembled cohesion metric. Additionally, seven general cohesion types are derived from an examination of previous measures and assumptions on the influence of cohesion on the software maintainability are claimed. Although a theoretical validation is mentioned, they do not provide any results. Furthermore, an empirical validation of the measures and assumptions is still missing.

Coupling Measures for Other Paradigms

As mentioned before, Stevens et al. (1974) were the first to define cohesion on an ordinal scale including coincidental, logical, temporal, procedural, communicational, sequential, and functional cohesion. Functional is the strongest and coincidental the weakest type of cohesion, which is generally determined by the associations between all pairs of elements.

Chidamber et al. (1994) introduced the very popular and repeatedly refined measure Lack of Cohesion in Methods (LCOM). This measure basically examines the sharing of attributes between methods and, thus, their supposed degree of similarity. They further claim that a “lack of cohesion implies [that] classes should probably be split into two or more subclasses”.

An own interpretation, as well as an improvement of the LCOM metric was provided by Hitz et al. (1995), as they identified access methods to cause an artificial decrease of cohesion in the original LCOM. Access methods usually reference one attribute and provide it to other methods, which accordingly even may not need to access any attributes directly and, thus, are isolated vertices in the graph.

For software parts, Briand, Morasca and Basili (1999b) propose the Ratio of Cohesive Interactions (RCI) measure and several variations of it. Based on a high-level design, they map data declarations (e.g. types, variables and constants), subroutines and interactions between them (DD-interaction, DS-interaction) onto a graph. Thereupon, the introduced measures basically consider the ratio between actual and possible connections. Additionally, they provide a theoretical and empirical validation of the new measures.

Allen et al. (1999) express cohesion based on their previously introduced intramodule coupling metric. Intramodule coupling and cohesion are considered measures of different attributes of relationships within modules. Cohesion is claimed to be the normalized ratio of the intramodule coupling values of the actual graph and a complete graph.

A new measure, which can be automatically calculated from the implemented code, was developed by Chae, Kwon and Bae (2000). For the Cohesion Based on Member Connectivity (CBMC) measure, they generate a graph-theoretical representation of the class structures extracted from the existing code and identify special methods etc. A software tool, named HYSS, provides this functionality for C++ programs. Additionally, the measure is theoretically and empirically evaluated against various other metrics.

Another technical feature, Vitharana et al. (2004) consider in their component design approach, is the cohesion value of a structured domain. The measure is calculated equally to the previously described coupling metric, except that it adds up coupling values of class-connections running within, instead of between, components. Again, no details on the utilized types of connections are provided.

As an extension to their coupling measures, Gui et al. (2007) also propose a new cohesion measure. Based on the definition of method similarity as a ratio of common and aggregate variables, they suggest the overall cohesion measure to be the normalized sum of pairwise method similarity.

Yu, Li, Zhao and Dai (2008) claim different cohesion metrics specifically designed for CO software. Based on a component dependence graph, which represents relations between the components operations and interfaces, they define cohesion between input-interfaces and operations, between operations and output-interfaces, as well as among operations. As the CO and SO design have several similarities, it is unfortunate that a theoretical or empirical validation of the measures is still missing.
Table 3. Classification of cohesion measures

CONCLUSIONS AND IMPLICATIONS FOR RESEARCH AND PRACTICE

The determination of an optimal granularity level for services is an important topic in the current SOC research focus. Coupling and cohesion metrics are common practice to assess the granularity of services. However, few approaches on SOC specific measures have been published so far. In this paper we have summarized the state of the art in coupling and cohesion measures for SOC, as well as established approaches for various other paradigms. Based on a literature survey, we derived a classification schema, which was utilized for an in-depth categorization of the measures. Several implications for practice and academia arise from the analysis of it.

For practice, Table 2 and Table 3 can be used to identify possible candidates for a specific application. In first place, it is preferable to choose a specialized measure for SOC. Furthermore, the conceptual foundations, for example the type and domain of a measure, provide useful information on the prerequisites. The types of nodes and connections allowed in the underlying model reveal the information necessary for the measure. Finally, measurement characteristics like the development phase and the level of validation show when the measure becomes usable and how it was reviewed. Additionally, for practice it would be relevant to implement coupling and cohesion measures in the design phase of a SOC development lifecycle to predict important design characteristics (e.g. maintainability) in an early state.

Various implications also arise for future research in the area of granularity measures for SOC. First of all, the number of SOC specific approaches is still limited. While their availability for older paradigms is rather high, it might in fact be advisable to examine existing metrics more closely. Promising OO measures should be adopted to the service-oriented paradigm (e.g. RCI (Briand et al., 1999b) or CBO (Chidamber et al., 1994)). Furthermore, as the concepts of component-oriented programming and SOC are already closely related, existing CO measures (e.g. Yu et al. (2008)) might only need some refinement to fit the new requirements. Once more measures exist, an extension of the proposed classification schema to a unified framework similar to Briand et al. (1998, 1999a) seems to be reasonable. Further research gaps become apparent from the above classification tables, as for example an empirical evaluation for most approaches is still missing and, additionally, important details as indirect connections and the inclusion of weights are left unconsidered so far. Especially the latter ones are already commonly considered in service identification approaches. On the other hand, the development of a unified service identification procedure, which can include arbitrary coupling and cohesion measures through a black-box principle, would be a promising approach. This can be based on a general graph or model representation, which might be entered by hand or generated from code through appropriate tools. This could also build up the basis to apply various coupling and cohesion measures to judge the quality of SOC implementations. Overall, a good amount of research still needs to be allocated to the development of coupling and cohesion measures and their applications.
REFERENCES


