A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software

Mikhail Perepletchikov, *Member*, *IEEE*, and Caspar Ryan

Abstract—One of the goals of Service-Oriented Computing (SOC) is to improve software maintainability as businesses become more agile, and thus underlying processes and rules change more frequently. This paper presents a controlled experiment examining the relationship between coupling in service-oriented designs, as measured using a recently proposed suite of SOC-specific coupling metrics and software maintainability in terms of the specific subcharacteristics of analyzability, changeability, and stability. The results indicate a statistically significant causal relationship between the investigated coupling metrics and the maintainability of service-oriented software. As such, the investigated metrics can facilitate coupling related design decisions with the aim of producing more maintainable service-oriented software products.

Index Terms—Services systems, design concepts, maintainability, product metrics, empirical studies.

1 Introduction

This paper is concerned with the impact of design level coupling on the maintainability of software products developed using the Service-Oriented Computing (SOC) approach. Specifically, a controlled experiment was conducted to evaluate the effectiveness of selected SO coupling metrics from Perepletchikov et al. [35] in predicting maintenance effort.

SOC is an emerging and promising software development paradigm which is based on the principle of encapsulating application and business logic within self-contained and stateless software *services*. Systems created using the SOC approach, that is, Service-Oriented (SO) systems, typically incorporate a large number of business processes that require frequent modification in order to facilitate rapid changes to the underlying business logic and rules [17], [34]. To this end, producing highly maintainable software is one of the main goals of SOC.

In previous software development paradigms, such as Procedural and Object-Oriented (OO) development, it was shown that software maintainability can be predicted, and consequently improved, early in the Software Development Lifecycle (SDLC) by quantifying the structural properties of software designs, such as *coupling* and cohesion, using dedicated metrics [12], [21]. Early prediction of maintainability is desirable given that software maintenance has long been regarded as one of the most resource-consuming development phases. For example, Pressman [39] and Zuse [48] suggest that over 60 percent of the total lifetime cost of a system is spent on maintenance.

Presently, little research effort has been dedicated to considering how the structural properties of service-oriented software designs may influence the maintainability of final software products. More significantly, there is a lack of mature, *empirically evaluated* SO-specific software metrics for measuring design properties in an automated and objective manner.

In previous work, Perepletchikov et al. [35] proposed a suite of 17 SO coupling metrics, which is the first effort of its kind for predicting maintainability of software in the early stages of the development process. The metrics were defined in terms of a formal model of service-oriented design [36] and theoretically validated using the property-based framework of Briand et al. [6]. Given that the aim of the original publication [35] was to formally define a suite of theoretically valid metrics, the empirical evaluation of the metrics was left to future work. Such evaluation is needed in order to establish empirically the relationship between the metrics and the quality characteristics they purport to predict, thereby showing the usefulness of metrics in practice [8], [25], [40].

This paper presents a controlled experiment conducted to evaluate the coupling metrics of Perepletchikov et al. [35]. The experiment involved a group of 10 participants, who were asked to perform a number of corrective and perfective maintenance activities on the controlled SO software system, developed specifically for this study. The system was composed of a number of distinct and self-contained services which exhibited different types and levels of coupling as reflected by the metrics (*independent variables*). The modification process was documented and measured, using standardized ISO/IEC maintainability metrics (*dependent variables*).

The correlation between the independent and dependent variables was then evaluated, using statistical approaches.

• *The authors are with the School of Computer Science and IT, RMIT University, Melbourne, Australia.*
E-mail: {mikhail.perepletchikov, caspar.ryan}@ rmit.edu.au.

Manuscript received 21 July 2009; revised 16 Feb. 2010; accepted 13 May 2010; published online 26 May 2010.
Recommended for acceptance by H. Muller.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2009-07-0180.
Digital Object Identifier no. 10.1109/TSE.2010.61.

The majority of the obtained results were in line with our hypotheses and expectations, indicating a statistically significant relationship between coupling in SO design artifacts and their maintainability. This, in turn, provides evidence of the potential usefulness of the evaluated SO coupling metrics, and serves as a motivator for future empirical studies on the structural property of coupling in the context of SOC.

2 SOA AND SOC—KEY CONCEPTS AND DEFINITIONS

Service-Oriented Architecture (SOA) is an abstract architectural model that describes how distributed service-oriented systems are composed (i.e., orchestrated or choreographed) to fulfill a specific domain or business requirement [3], [34]. At a conceptual level, SOA consists of three primary entities: 1) service providers, who publish service descriptions and realise software services, 2) service consumers, who discover service descriptions, and invoke services, and 3) service registries or repositories (such as UDDI [47]) that maintain a directory of services.

Services in SOA are commonly treated as technology-agnostic "black-boxes," where corresponding *service interfaces* constitute the only visible part of the software architecture [18]. That is, SOA prescribes a conceptual architecture that advocates the principles of *loose coupling* without enforcing constraints on the actual design and implementation of services and the individual service-oriented systems. Loose coupling in SOA is achieved by the logical and physical separation of service consumers from service providers via service registries, meaning that there is no need for preexisting relationships between parties.

While SOA embodies an overall conceptual architecture, Service-Oriented Computing (SOC) is the concrete software development paradigm covering the process of developing software applications structured in terms of autonomous software services that encapsulate well-defined business functions [17]. One of the main goals of SOC is to provide systematic guidance for the identification and realization of services at the detailed design and implementation levels [33]. To this end, *the main focus of the research presented in this paper is the design phase of SOC.*

Fig. 1 illustrates a design view of an SO system, where *services* consist of two types of fundamental design artifact: *service interfaces* and *service implementation elements* (that realize operations exposed in a service interface). There are no technological constraints on the paradigms and languages used to define service interfaces (although WSDL is commonly used to describe service interfaces in present implementations [47]) nor realize service implementation elements, although a typical example of such would be Java or C++ classes, as shown in Fig. 1.

There are two important design-level characteristics of SOC that differentiate it from the previous paradigms such as OO and Procedural development [36], [38]:

SOC introduces an additional level of abstraction. The Procedural paradigm has only one main level of design abstraction: a *procedure*. The OO paradigm operates on two levels of design abstraction where *OO methods* are

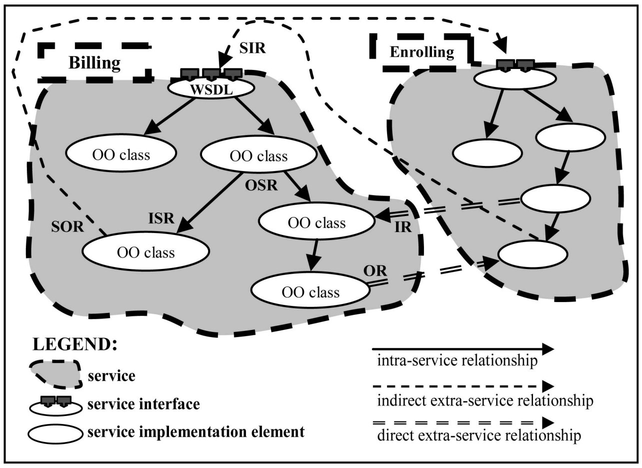
Billing
SIR
Enrolling
WSDL
OO class
OO class
OSR
SOR
ISR
OO class
IR
OO class
OR
OO class
LEGEND:
service
service interface
service implementation element
intra-service relationship
indirect extra-service relationship
direct extra-service relationship

Fig. 1. Example SO design showing different relationship types.

encapsulated within *OO classes*. In contrast, the SOC paradigm introduces a third level of abstraction and encapsulation: a *service*. In service-oriented systems, *operations* (e.g., OO methods) are aggregated into *implementation elements* (e.g., OO classes) that implement the functionality of a ***service*** as exposed through its *service interface.*

A service interface is an important first-class design artifact. Correctly identifying service interfaces is a challenging and important design activity. This is because service-oriented systems should be structured in terms of independent and self-contained services, with service interfaces being the primary entry points of a system [16], [24].

The above characteristics suggest the need to quantify coupling in SOC, based on the locality of coupling relationships from the service boundary perspective, where the concept of *loose coupling* refers to the absence of direct implementation-level dependencies between system services. Specifically, three distinct coupling relationship types (shown in Fig. 1) should be considered in SO designs [35]: 1) *incoming intraservice* relationships (ISR) and *outgoing intraservice* relationships (OSR), 2) *incoming indirect extra-service* relationships (SIR) and *outgoing indirect extraservice* relationships (SOR), and 3) *incoming direct extraservice* relationships (IR) and *outgoing direct extraservice* relationships (OR). Section 5 provides a detailed description of the SO coupling relationship types.

3 INTERNAL AND EXTERNAL QUALITY CHARACTERISTICS

Developing high quality software should be one of the main targets of any software engineering process independent of the development paradigm in use [20]. In software engineering, quality characteristics are typically categorized as *internal* or *external*. That is, the design of any software product is said to possess a number of measurable *internal* characteristics or structural properties (e.g., coupling, cohesion, and size) that have a causal effect on *external* quality characteristics, such as *maintainability*.

According to the *ISO/IEC 9126-1:2001* [26] standard, the quality characteristic of maintainability can be defined as *the level of effort required for modifying the software product,* and can be subdivided into four measurable subcharacteristics: analyzability, changeability, stability, and testability. These subcharacteristics are defined in [26] as:

- *Analyzability*: the capability of the software product to be diagnosed for causes of failures or for the parts to be modified to be identified.
- *Changeability*: the capability of the software product to enable specified modifications to be implemented.
- *Stability*: the capability of the software product to avoid unexpected effects from modifications.
- *Testability*: the capability of the software product to enable modified software to be validated.

Additionally, maintenance tasks can be categorized according to the nature of the change request. For example, the ISO/IEC/IEEE maintenance standard [27] states that a modification request can be classified as either a correction (*corrective* and *preventive* activities) or enhancement (*adaptive* and *perfective* activities). To this end, the present work is concerned solely with *corrective* and *perfective* maintenance[1] since these activities are highly relevant to SOC. This is because SO software products typically include a large number of business rules and associated business processes, which have been shown to be the most unstable part of software applications [44]. This suggests a potential increase in the rate and number of corrective and perfective tasks required to keep up with rapidly changing business requirements [17].

4 The Structural Property of Coupling

It was shown in previous work that structural properties of software designs can influence some of the subcharacteristics of maintainability [12], [23], [29]. For example, a number of empirical studies demonstrated that the structural property of *coupling* is strongly correlated to the analyzability, changeability, and stability of software products,[2] regardless of development paradigm [9], [15].

The concept of coupling was originally defined for procedural systems by Stevens et al. as *"the measure of the strength of association established by a connection from one module to another"* [43, p. 233], where coupling was classified based on the type of connection—data or control. The authors later extended their definition [46] in order to characterize four major factors that influence coupling:

1. type of connection between modules,
2. complexity of the interface,
3. type of information flow (data or control), and
4. binding time of connection.

The notion of coupling was then extended for OO systems due to the existence of additional mechanisms that can influence coupling, such as polymorphism. Also, there are two main subjects of interest in OO design, namely, classes and methods, as opposed to procedural systems where the procedure (a module consisting of code statements) is the main subject of interest. To this end, coupling in OO was defined as *"the interdependency of an object on other objects in a design representing the count of other objects that would have to be accessed by a given object in order for that object to function correctly"* [4, p. 7].

It is important to note that coupling can be measured at different levels of abstraction, ranging from *high-level design* through to *executable implementations*, with the target level of abstraction influencing the definition of metrics and their measurement procedure. Ideally, metrics should be collected as early in the SDLC as possible since the sooner problems in the software structure can be identified, the less effort required to correct them.

4.1 Coupling in the Context of SOC

The structural property of coupling is yet to be thoroughly investigated in the context of SOC. To this end, Perepletchikov et al. [35] redefined the notion of coupling in order to address the specifics of service-orientation (such as the inclusion of an additional level of design abstraction—a *service,* refer to Section 2). More importantly, a suite of theoretically-validated coupling metrics, defined according to the formal model of SO design [36], was proposed to quantify coupling at three different levels of design abstraction: 1) *system* level, 2) *service* level, and 3) service *implementation element* level. Furthermore, the element level metrics were designed to quantify three distinct types of coupling relationships, shown in Fig. 1:

1. *intraservice* relationships between implementation elements belonging to the same service—ISR and OSR relationships in example SO design in Fig. 1,
2. *indirect extraservice* relationships between implementation elements of a given service and service interfaces of other (external) services—SIR and SOR relationships in Fig. 1,
3. *direct extraservice* relationships between implementation elements belonging to different services—IR and OR relationships in Fig. 1.

These relationships were assumed to have a varying impact on the maintainability of SO software, and thus the corresponding metrics were initially weighted (using a coarse approximation) with respect to one another, based on their perceived influence on maintainability.

Note, however, that direct extraservice relationships should be avoided since they violate one of the key principles of SOC, service autonomy [32]. Nevertheless, most existing SO applications incorporate legacy components which were not designed according to the principles of SOC and, as such, direct extraservice relationships are common in practice [17]. To this end, the corresponding metrics are useful since they can quantify nonconformance to SO principles.

5 SO Coupling Metrics

The metrics presented in this section and empirically evaluated in Section 6 were originally published in [35] along with a formal set-theoretic definition of each metric, and a description of the corresponding theoretical validation process. Of the 17 metrics, only the *element-level* metrics

1. *Corrective* maintenance refers to modifications necessitated by errors (i.e., defects) in a software product [27]. *Perfective* maintenance refers to modifications performed to provide new functionality or improvements for users, or reverse engineering to improve performance or maintainability [27].
2. Testability typically refers to the existence of inbuilt testing functions, and is not related to the structure of software.

(six metrics in total) are covered in this paper. The service level metrics, which represent an aggregation of the element-level metrics, are not evaluated since the evaluation of their constituent metrics is sufficient to indirectly establish the empirical validity of the aggregate metrics. Furthermore, the system level metrics are not evaluated since developing multiple full system designs as required for their evaluation is beyond the scope of this paper.

The metrics are designed to measure coupling based on two different factors: direction of coupling communication, and the three different SO coupling relationship types described in Section 4.1. The metrics can be collected by examining design documents related to the structure and behavior of an SO system including UML class, sequence and collaboration diagrams, flow charts or data flow diagrams, business process scripts, etc. In addition, business level information captured in workflows and use-case diagrams can be used to calculate metrics in terms of the participation of external system actors, in cases where more explicit structural information is not available. This is particularly relevant to extraservice incoming coupling, where it might be difficult to determine the concrete structural characteristics of clients located outside known system boundaries.

5.1 Intraservice Coupling

This type of coupling is comparable to the notion of coupling in Procedural and OO designs. This is because an individual service can be considered as a Procedural or OO subsystem when investigated in isolation from the other services in the system; therefore, the impact of high intraservice coupling on maintainability is expected to be similar to that suggested for Procedural/OO systems [15], [22]. More specifically, high intraservice coupling is assumed to influence the specific maintainability characteristics in a distinct way [35] as follows:

- High *incoming* intraservice coupling can influence service *changeability* and *stability*.
- High *outgoing* intraservice coupling can influence the *analyzability* and *changeability* of an element.

The above assumptions will be formally redefined as experimental hypotheses (Section 6.1.3, Hypotheses $H_{coup1.1}$ and $H_{coup1.2}$) and statistically tested in Section 6.2.2.

The following two metrics are designed to measure incoming and outgoing intra-service coupling.

Weighted intraservice incoming coupling of an element (WISICE). WISICE for a given service implementation element e, belonging to a service s, is the count of the number of other implementation elements of the same service s coupled to element e (ISR relationship type in Fig. 1).

Motivation: Services are intended to be independent components, and thus can be maintained in isolation from the system. Therefore, it is useful to measure the coupling within a single service. Specifically, it is expected that high intraservice coupling of one or more service implementation elements can indicate a potential design problem that should be fixed prior to commencing the implementation of the service.

Weighted intraservice outgoing coupling of an element (WISOCE). WISOCE for a given service implementation element e, belonging to a service s, is the count of the number of other implementation elements of the same service s that are used by element e (OSR relationship in Fig. 1).

Motivation: As described previously, services are intended to be independent components which can be maintained in isolation. As such, it is useful to measure the coupling within a single service.

5.2 Indirect Extraservice Coupling

Indirect extraservice coupling concerns relationships that occur through a service interface (i.e., not *directly* to service implementation elements), thereby supporting the notion of service-autonomy and loose coupling, as advocated by SOC literature [1], [3], [33].

Nevertheless, according to [35], indirect extraservice coupling should still be *weighted higher than intraservice coupling* because when system functionality is encapsulated in different services, which can be situated across various logical and physical boundaries, maintenance effort is expected to increase accordingly. This will be tested in Section 6.2.3.

Also, as was the case with intraservice coupling, the direction of communication (or *locus of impact* [7]) is expected to influence the specific subcharacteristics of maintainability as reflected by the following two metrics.

Extraservice incoming coupling of service interface (ESICSI). ESICSI for a given service s is a count of the number of system elements not belonging to service s that couple to this service through its interface si_s (SIR relationship in Fig. 1).

Motivation: Although indirect extraservice coupling can be considered as a desirable form of loose coupling, it should still be minimized where possible. This is because, as the value of ESICSI increases, the service becomes more critical from a systemwide perspective, thereby resulting in a decrease in *system changeability* and *stability*. This will be tested in Section 6.2.2 according to Hypothesis $H_{coup1.3}$.

Element to extraservice interface outgoing coupling (EESIOC). EESIOC for a given service implementation element e is a count of the number of other service interfaces in the system that are used by (coupled to) element e (SOR relationship in Fig. 1).

Motivation: As argued previously, it is advisable to avoid excessive and unnecessary indirect extraservice coupling. As the value of EESIOC for a given implementation element increases, so does the dependency of this element on the other services in the system. As such, the *analyzability and changeability of an element* is likely to be affected, given the greater effort needed to analyze (and change) an element using functionality provided by many external artifacts. This will be tested in Section 6.2.2 (Hypothesis $H_{coup1.4}$).

5.3 Direct Extraservice Coupling

This type of coupling covers the direct relationships between implementation elements belonging to different services. According to [35], direct extraservice coupling can be considered as the strongest (least desirable) type of extraservice coupling because it breaks service autonomy [17] via explicit dependencies between service implementations, thus potentially decreasing their reusability and

TABLE 1
Summary of the Investigated SO Coupling Metrics

METRIC	TYPE OF COUPLING	LOCUS OF IMPACT	WEIGHT
WISICE	intra-service	incoming	1
WISOCE		outgoing	
ESICSI	indirect extra-service	incoming	2
EESIOC		outgoing	
WESICE	direct extra-service	incoming	3
WESOCE		outgoing	

substitutability [42]. To this end, Perepletchikov et al. [35] argue that this type of coupling should be *weighted higher than both intraservice and indirect extraservice coupling* types. This will be tested in Section 6.2.3. Also, the direction of communication is again expected to influence the specific subcharacteristics of maintainability as reflected by the following two metrics:

Weighted extraservice incoming coupling of an element (WESICE). WESICE for a given service implementation element e of service s is a count of the number of system elements not belonging to service s that couple to element e directly (IR relationship shown in Fig. 1).

Motivation: Direct extraservice coupling introduces tight (implementation-dependent) coupling between services and should therefore be avoided. Specifically, incoming coupling from service implementation elements belonging to different services, to element e, is expected to negatively influence *changeability and stability of a system* since as WESICE increases, so does the number of external elements and services dependent upon the implementation characteristics of element e. This will be tested in Section 6.2.2 (Hypothesis $H_{coup1.5}$).

Weighted extraservice outgoing coupling of an element (WESOCE). WESOCE for a given service implementation element e of service s is the count of the number of system elements not belonging to the same service that are used by (coupled to) this element (OR relationship in Fig. 1).

Motivation: As the value of WESOCE for a given element increases so does the (tight) implementation-level dependency of this element on the other implementation elements in the system. As such, the *analyzability and changeability of this element* will be affected. This will be tested in Section 6.2.2 (Hypothesis $H_{coup1.6}$).

5.4 Metrics Summary

Table 1 provides a summary of the metrics investigated in this paper, where the values in the *relative weight* column are intended to represent the hypothesised strength of impact of different coupling types on the analyzability, stability, and changeability of SO software, as explained in the metric motivation sections above. In order to maintain the integrity of the metrics from the measurement theory perspective, such weights need to be defined on a ratio scale. However, since this requires large-scale empirical studies, this is left to future work. As such, the weighting values of 1, 2, and 3 are a coarse approximation provided for illustrative purposes only.

6 Empirical Study[3]

The empirical study (or controlled experiment) presented in this section is designed to evaluate the relationship between the investigated coupling metrics and the specific subcharacteristics of maintainability (i.e., analyzability, stability, and changeability) they purport to predict. Specifically, there are two main goals of this study, defined according to the GQM framework of Basili et al. [5]:

GOAL-COUP1. Evaluate the *predictive capability* of selected service-oriented coupling metrics with respect to early estimation of the analyzability, stability, and changeability of SO software, from the point of view of software engineers, in the context of an experimental SO software system.

- This will be done by comparing the maintainability of lowly and highly-coupled elements for each distinct coupling relationship type (i.e., incoming and outgoing: intraservice, indirect extraservice, and direct extraservice coupling).

GOAL-COUP2. Evaluate the *relative impact (strength)* of selected service-oriented coupling metrics on the analyzability, stability, and changeability of SO software, from the point of view of software engineers, in the context of an experimental SO software system.

- This will be done by comparing the maintainability values collected for the highly coupled elements for each distinct coupling relationship type. This, in turn, should allow: 1) ranking different coupling relationship types and corresponding metrics based on their influence on maintainability, and 2) comparing SO metrics with the widely accepted OO metric, Coupling Between Objects (CBO) [11]. Such a comparison is possible because CBO is structurally equivalent to the WISCE metric (refer to Section 5.1) intended to measure intraservice relationships.

An empirical evaluation of software metrics can be achieved using either targeted (controlled) experiments conducted under research settings, or case studies of industrial software products [8]. Both strategies can have a contrasting impact on the *internal* and *external* validity [8] of the produced results. More specifically, targeted experiments provide greater support for controlling instrumentation effects [9] that can influence the internal validity of the study, but can negatively affect external validity since such experiments are not necessarily representative of real-world practice insofar as they are subject to time and scope constraints, which in turn impose limitations on the size of experimental artifacts and tasks. In contrast, case studies maximize external validity, but make it more difficult to control instrumentation effects.

3. This study was conducted as part of a comprehensive investigation of two important structural properties of SO software, coupling and cohesion. Although the empirical study conducted to evaluate the impact of cohesion on maintainability [38] was conducted independently of the coupling study presented in this paper, the studies share some common features. In particular, the same participants (refer to Section 6.1.1) were employed in both studies, and the overall study design (Section 6.1.6) and experimental procedure (Section 6.1.7) were comparable. Moreover, both studies used similar techniques to alleviate possible threats to validity (Section 6.4).

The study presented in this section follows the first approach since it has been suggested that the internal validity of the study should be determined prior to establishing external validity [45]. Additionally, SOC is an emerging paradigm, and as such it is difficult to conduct large-scale industrial investigation at this stage.

6.1 Experimental Protocol

This section presents the experimental protocol that was followed when performing the experimental tasks and analyzing the results. Note that an effort has been made to provide a comprehensive description of the protocol so that the study can be replicated in future work.

6.1.1 Participants

The study employed 10 subjects: 1) five industry practitioners and 2) five postgraduate research students undertaking their study in the School of Computer Science and IT, RMIT University, Melbourne, Australia. *Purposive sampling* [41], which is a participant selection technique frequently used in behavioral sciences, was employed to select participants with comparable experience with software development and maintenance and knowledge of various development paradigms, including SOC and OO. The level of experience and knowledge of each participant was evaluated using a User Profile questionnaire and a pretest task. All participants were male in the 23-29 age group, possessed OO development and maintenance experience ranging from six months to five years, and had some basic experience with SO development. Also, all participants were unpaid volunteers who had professional interest in SOC.

6.1.2 Dependent Variables

The dependent variables were taken from the ISO/IEC TR 9126-1:2001 [26] suite of maintainability metrics. Note that all ISO/IEC metrics are defined on a ratio scale, thereby being suitable for the parametric statistical analyses (such as one-way ANOVA) used in this study.

Analyzability Metric. Failure Analysis Efficiency $(\mathrm{FAE}) = \mathrm{Sum}\ (\mathrm{T})/\mathrm{N}$, *where* $\mathrm{T} = \mathrm{time}$ taken to analyze each cause of failure (or time taken to locate a software fault), and $\mathrm{N} = \mathrm{number}$ of failures, of which causes are found. *The interpretation of possible values*: $0 < \mathrm{FAE}$; the closer to 0, the better.

Changeability Metric. Modification Complexity $(\mathrm{MC}) = \mathrm{Sum}(\mathrm{T})/N$, *where* $\mathrm{T} = \mathrm{work}$ time spent on each change and $\mathrm{N} = \mathrm{number}$ of changes. *The interpretation of possible values:* $0 < \mathrm{MC}$; the closer to 0, the better.

Stability Metric. Modification Impact Localization $(\mathrm{MIL}) = \mathrm{A}/\mathrm{B}$, *where* $\mathrm{A} = \mathrm{number}$ of emergent adverse impacts (failures) in the system after modifications, and $\mathrm{B} = \mathrm{number}$ of modifications made. *The interpretation of possible values*: $0 <= \mathrm{MIL}$; the closer to 0, the better.

6.1.3 Experimental Hypotheses

Two distinct sets of hypotheses were defined according to the goals of this study and tested for statistical significance in Section 6.2.2.

The first set of hypotheses (Table 2, $\mathrm{H}_{\mathrm{coup1.1}}$–$\mathrm{H}_{\mathrm{coup1.6}}$) is related to GOAL-COUP1, based on the assumption that a highly-coupled SO design element will have a significantly negative impact on the maintainability subcharacteristics compared to a lowly-coupled element. For example, Hypothesis $\mathrm{H}_{\mathrm{coup1.1}}$ can be interpreted as:

An element with a high[4] *value of the Weighted Intraservice Incoming Coupling between Elements (WISICE) metric will result in significantly lower (or worse) service changeability*[5] *(as reflected by a higher value of the Modification Complexity (MC) metric), and stability (i.e., a higher value of the Modification Impact Localization (MIL) metric) compared to an element with a low value of WISICE.*

The second set of hypotheses (Table 2, $\mathrm{H}_{\mathrm{coup2.1}}$–$\mathrm{H}_{\mathrm{coup2.6}}$) is based on the assumption that different coupling relationship types have varying impact on the maintainability subcharacteristics according to GOAL-COUP2. For example, Hypothesis $\mathrm{H}_{\mathrm{coup2.2}}$ can be interpreted as:

An element with a high value of the Element to the Extraservice Interface Outgoing Coupling (EESIOC) metric will have a more negative effect on analyzability (i.e., greater Failure Analysis Efficiency (FAE)) and changeability (i.e., greater Modification Complexity (MC)) compared to an element with the same (high) value of Weighted Intraservice Outgoing Coupling between Elements (WISOCE).

Note that the hypotheses presented in Table 2 are stated as *alternative hypotheses*, where the corresponding *null hypotheses* would be defined in terms of the absence of examined effects. Moreover, although all hypotheses were defined in terms of multiple dependent variables in order to minimize the total number of hypotheses and thus improve the readability of this section, they are tested individually for each stated dependent variable.

TABLE 2
Experimental Hypotheses—Details

	HYPOTHESIS CONDITION	*AFFECTED DESIGN ARTEFACT*
$\mathrm{H}_{\mathrm{coup1.1}}$	↑WISICE* => ↑MC** and ↑MIL	service
$\mathrm{H}_{\mathrm{coup1.2}}$	↑WISOCE => ↑FAE and ↑MC	element
$\mathrm{H}_{\mathrm{coup1.3}}$	↑ESICSI => ↑MC and ↑MIL	system
$\mathrm{H}_{\mathrm{coup1.4}}$	↑EESIOC => ↑FAE and ↑MC	element
$\mathrm{H}_{\mathrm{coup1.5}}$	↑WESICE => ↑MC and ↑MIL	system
$\mathrm{H}_{\mathrm{coup1.6}}$	↑WESOCE => ↑FAE and ↑MC	element
$\mathrm{H}_{\mathrm{coup2.1}}$	[↑ESICSI > ↑WISICE] => ↑MC and ↑MIL	system
$\mathrm{H}_{\mathrm{coup2.2}}$	[↑EESIOC > ↑WISOCE] => ↑FAE and ↑MC	element
$\mathrm{H}_{\mathrm{coup2.3}}$	[↑WESICE > ↑WISICE] => ↑MC and ↑MIL	system
$\mathrm{H}_{\mathrm{coup2.4}}$	[↑WESOCE>↑WISOCE] => ↑FAE and ↑MC	element
$\mathrm{H}_{\mathrm{coup2.5}}$	[↑WESICE > ↑ESICSI] => ↑MC and ↑MIL	system
$\mathrm{H}_{\mathrm{coup2.6}}$	[↑WESOCE >↑EESIOC] => ↑FAE and ↑MC	element

Legend: *↑ high value;* ***=>*** *results in (affects);*
[↑X > ↑Y] => ↑A and ↑B *– a high value of independent variable X results in a larger increase (i.e. has more negative affect) of dependent variables A and B compared to a high value of independent variable Y.*

** the dependent/independent variables are described in Section 5 and 6.1.2.*

*** high values of FAE, MC, and MIL ISO/IEC metrics indicate low (or worse) analysability, changeability and stability respectively.*

4. The actual values used to represent high (and low) coupling in all coupling hypotheses are described in Section 6.1.4.

5. Incoming coupling is related to changeability and stability, whereas outgoing coupling is related to changeability and analyzability as discussed in Section 5.

6.1.4 Experimental Material

The experimental material consisted of a controlled service-oriented software system developed specifically for this study and designed to support the investigation of the experimental hypotheses presented in the previous section. The system was based on an existing prototypical service-oriented Academic Management System (AMS), which was loosely modeled on the rules and procedures common to RMIT University. The original system was designed to provide support for the practical assessments used in a number of courses run by the School of the Computer Science and IT, RMIT University. Note that the chosen application domain (educational organization) had the advantage of being easily comprehensible by the participants, thereby ensuring that system requirements could be easily interpreted.

The experimental version of AMS was composed of *three distinct services* implemented using Java EE 5 platform, and exposed via both local and WSDL-based interfaces. Each service was located in a dedicated package, as shown in Appendix A, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2010.61, with further implementation details available in the online appendix [37].

In total, three services were required in order to support the controlled investigation of different types of coupling as follows:

1. Service **SER-COUP1** (ams.services.academic-management service shown in Appendix A) was designed to support the investigation of *intraservice* coupling (hypotheses $H_{coup1.1}$, $H_{coup1.2}$, and $H_{coup2.1}–H_{coup2.4}$).
2. Service **SER-COUP2** (ams.services.student-management service from Appendix A) was designed to support the investigation of *indirect extraservice* coupling (hypotheses $H_{coup1.3}$, $H_{coup1.4}$, $H_{coup2.1}$, $H_{coup2.2}$, $H_{coup2.5}$, and $H_{coup2.6}$).
3. Service **SER-COUP3** (ams.services.enrollment-support service shown in Appendix A) was designed to support the investigation of *direct extraservice* coupling (hypotheses $H_{coup1.5}$, $H_{coup1.6}$, and $H_{coup2.3}–H_{coup2.6}$).

Additionally, each of the experimental services was logically divided into two subservices in order to support the investigation of the incoming (SER-COUP1a, SER-COUP2a, and SER-COUP3a) and outgoing (SER-COUP1b, SER-COUP2b, and SER-COUP3b) coupling relationships in isolation from one another.

Finally, the experimental system included a number of utility services that provided basic support for data access and manipulation, and communication with external partners (prototype web services) running on a different application server. Such utility services were required in order to remove any potential influence of the specific technologies used to develop the experimental systems (for example EJB3 and *Java Persistence API*) on the maintenance efforts. Additionally, a number of support services that provided means for evaluating the extraservice coupling metrics were included. The participants were notified that all utility/support services were fault free.

The following characteristics were common to all three experimental services:

- Each service aims to evaluate the coupling-related hypotheses using four designated implementation elements/or service interfaces: two *lowly-coupled [incoming and outgoing]* elements/or interfaces, and two *highly -coupled [incoming and outgoing]* elements/or interfaces, where: 1) an implementation element/service interface is considered to be *lowly-coupled* if it is coupled to/from one implementation element or service interface only, and 2) an implementation element/service interface is considered to be *highly-coupled* if it is coupled to/from five other implementation elements or service interfaces. The number 5 was chosen because it allowed developing subsystems of manageable size, and also because it represents the maximum number of couples for a given implementation element in the original AMS.
- Each element designed to investigate low and high *outgoing coupling* contained one explicitly inserted fault in order to assess the analyzability of elements under study. All faults were related to the same problem domain, and their *cognitive complexity*[6] was approximately equivalent, as reflected by the output of a pilot study (refer to Section 6.1.5) and the feedback obtained from the participants.

Given our aim to conduct a controlled experiment, an effort was made to manipulate only the coupling of services while keeping the other properties (or *controlled variables*) as constant as possible in order to prevent their influence on the experimental results as follows:

1. *Interface Size and Complexity:* All three experimental services expose an equal number (four) of service operations in order to ensure their comparability in terms of interface size and complexity. The number 4 was chosen because it allowed developing services of manageable size, while maximizing the amount of experimental data. Additionally, it represents the mean number of service operations exposed in the interfaces of services included in the original AMS.
2. *Implementation Size*: The lines of code (LOC) measures for each of the experimental services were similar (300-350 LOC per service). Note that the total size of the system, including the supporting services, was approximately 3,500 LOC.
3. *Cohesion*: The cohesion of experimental services was held constant, with all three services having an average cohesion ranging from 0.47 to 0.52, as measured by the Total Interface Cohesion of a Service (TICS) metric derived and evaluated by the authors in prior work [38].

6.1.5 Experimental Tasks

The experimental tasks were intended to directly support the evaluation of the analyzability, changeability, and

6. Cognitive complexity can be defined in terms of the level of effort used by a person performing a given task, where the cognition processes common to the software maintenance tasks include reading code (chunking) and searching through code (tracing) [10].

stability subcharacteristics of maintainability in order to test the experimental hypotheses defined in Section 6.1.3.

TASK-COUP1. The participants were asked to *determine* the causes of six failures related to the core functionality of the system (one failure per outgoing lowly and highly-coupled element under study). An effort was made to ensure that all causes of failures (or software faults) were comparable in terms of their cognitive complexity. This task can be considered as a *corrective* maintenance activity, being designed to evaluate *analyzability* using the Failure Analysis Efficiency (FAE) metric. Participants were required only to identify the faults by noting the name of the faulty implementation element/s and operation/s, but not fix them.

TASK-COUP2. The participants were asked to implement 12 source-code changes to the business rules and logic—one change per designated incoming and outgoing [lowly/highly] coupled element, in each of the three experimental services. This task can be considered as *perfective* maintenance, and was intended to support the investigation of the *changeability* and *stability* subcharacteristics. Note that changes were related to specific functional requirements of the AMS system, as well as the particular implementation characteristics of each service, and as such, experience or learning effects [9] were unlikely to be a significant factor. Also, as with the introduced failures, an effort was made to ensure that all changes were conceptually (or cognitively) comparable.

A pilot study using two experienced software engineers (who were not participating in the main study) was conducted to evaluate the cognitive similarity of the introduced failures and required system changes, and to approximate the completion times for both tasks. Both software engineers completed all of the experimental tasks within the allocated time-frame of 3 hours, having ranked the cognitive complexity of all failures and changes as low. This was expected since the failures/changes were designed to have low complexity in order to reduce potential fatigue effects.

6.1.6 Experimental Design

The study was based on an *incomplete within-subjects* design [31], where each condition was administered to each participant only once, and the order of administering the conditions was varied systematically across participants. A within-subjects experimental design was considered suitable for this study because it requires fewer participants to maintain the statistical power of the experiments compared to a between-subjects design [14]. Furthermore, the error variance due to differences among participants is reduced in within-subjects designs [41], which is especially significant in the area of empirical software engineering, where potentially strong variation in participant capabilities and skills is a major concern [9].

Note, however, that one of the limitations of a within-subjects design is that the independent variables can become confounded with the order of presentation because of *practice effects* [41]. Practice effects can arise as the experimentation progresses, due to improvements in the participants' skills and knowledge (*learning effects* [9]), and degradation of participants' ability to perform experimental tasks (*fatigue effects*).

To deal with the practice effects in this study, thereby increasing its internal validity [45], the participants were exposed to the various levels of treatment using the *selected orders approach* [41], where only a subset of all possible orders is used, with the number of selected orders equaling some multiple of the number of conditions in the experiment. Specifically, the *random starting order with rotation* technique was employed to ensure that each condition appears in each ordinal position equally often. This technique involves choosing a random initial order (or sequence) of the conditions, which is then systematically rotated with each condition moving one position (in this case to the left) on each rotation [41].

6.1.7 Experimental Procedure

The experiment consisted of two separate phases: the pre-experimental phase and the actual experiment.

During the pre-experimental phase, the participants were given one week to familiarize themselves with the System Requirements Specification (SRS) document for AMS. Next, all participants attended a 30 minute face-to-face group tutorial session, where they had a chance to ask questions related to the requirements of AMS. Furthermore, at the end of the tutorial session, the participants were asked to rank their understanding of the functional requirements of the AMS system on a scale of 1 (low) to 5 (high). All participants indicated that they had a good understanding of these concepts (as reflected by the uniformly high rankings of 5).

The actual experiment was conducted in a controlled laboratory setting in order to eliminate confounding factors, such as unplanned distractions, that could affect the performance of participants. The PCs located in the lab had identical hardware and software configurations, with Eclipse 3.3 chosen as the implementation perform, since all participants had prior experience with this specific Integrated Development Environment (IDE). At the beginning of the experiment, the participants were given a document describing the tasks to be performed, and the details of the tasks were then described by one of the authors in a 15 minute question-answer session. Participants were instructed not to talk among themselves, but instead direct any questions to the monitor. All participants managed to complete the tasks within the allocated time-frame of 3 hours.

6.1.8 Data Collection Procedure

The duration of individual maintenance tasks was collected in real time. Specifically, the task commencement time was noted, and as soon as a given participant finished one of the tasks he (all participants were male, refer to Section 6.1.1) would indicate this to the monitor who would then record the completion time on a PDA. In addition to recording the duration of the maintenance tasks during the experimental runs, the modified systems were later examined and tested by the first author, using a suite of unit and integration tests, in order to collect the data needed to calculate the ISO/IEC stability metric Modification Impact Localisation (MIL). This was necessary since the participants were asked to implement the changes, but were not asked to test the experimental system after the changes were made (otherwise, it would be impossible to collect the values of MIL due to a lack of newly introduced and undiscovered faults).

The following data points were collected individually for [incoming/outgoing] lowly and highly-coupled implementation elements in each of the three experimental services on a per task basis as follows:

TASK-COUP1.

Failure Analysis Efficiency (FAE)

- Time (in minutes) taken to analyze the cause of failure in three experimental services for each pair of [outgoing] lowly and highly-coupled elements for every participant ($1 \text{ failure} \times 2 \text{ elements} \times 3 \text{ services} \times 10 \text{ participants} = 60$ data points).
- The correctness (number of failures of which causes are found) of the failure analysis described above (60 data points). The correctness was evaluated in terms of a Boolean value [true/false]. That is, the failure analysis was considered correct only when all causes of a given failure were found.

TASK-COUP2.

Modification Complexity (MC)

- Time (in minutes) taken to implement a change to the functional requirement in three experimental services for each pair of [incoming/outgoing] lowly- and highly-coupled elements for every participant ($1 \text{ change} \times 3 \text{ services} \times 4 \text{ elements} \times 10 \text{ participants} = 120$ data points).

Modification Impact Localization (MIL)

- Number of detected faults in the system after changes to the functionality described above for each pair of [incoming] lowly and highly-coupled elements for every participant ($1 \text{ change} \times 3 \text{ services} \times 2 \text{ elements} \times 10 \text{ participants} = 60$ data points).

6.1.9 Analysis Procedure

The collected experimental data were analyzed using the Analyze-it 3.0 [2] statistical software tool.

To begin, the Shapiro-Wilk test [28], which is a recommended normality test for smaller sample sizes (up to 1,000), showed that the distribution of both dependent variables did not deviate significantly from normality (all Shapiro-Wilk (W) values were greater than 0.75). This suggested that parametric tests were suitable for this study. Nevertheless, the significance probability values (p) for the tests were high (all $p > 0.3$), thereby suggesting that it is possible that the observed normality results are incorrect. As such, the decision was made to also use alternative nonparametric tests. In every case, the nonparametric test supported the findings of the corresponding parametric test.

Second, the *level of significance* (α), which reflects the probability of falsely rejecting a null hypothesis (Type I error), was set to the commonly used (scientific) level of 0.05. Note that it is not uncommon to use higher levels of α in software engineering experiments. For example, Briand et al. [9] suggest that an α value as high as 0.2 might be considered when making an informed decision regarding practical utility of software metrics, even though the empirical evidence is not strong enough to make a scientific statement with a high degree of confidence.

Finally, The G*Power 3 [19] software tool was used to conduct a power analysis in order to estimate the impact of effect size on the statistical tests used in the study. The analysis showed that given our: 1) sample size of 10 participants in a within-subjects design, 2) chosen level of $\alpha = 0.05$, and 3) a power value of 0.8[7]; the effect sizes required to achieve statistically significant results can be classified as large according to the categorization of Cohen [14] whereby, for a one-way ANOVA, the conventional effect sizes (f) are: small $f = 0.1$, medium $f = 0.3$, and large $f \geq 0.1$. As such, the effect sizes required to show significance in the present study range from 0.5 to 0.6, thereby suggesting that the statistical tests could produce a Type II error [14] (i.e., fail to reject a null hypothesis when it is in fact false) when the relationships involved do not exhibit large effect sizes.

6.1.10 Statistical Tests

A number of parametric and nonparametric statistical tests have been employed to evaluate the experimental hypotheses. The tests were chosen based on their suitability to the goals and design of the study. For example, the within-subjects experimental design adopted in the study requires the selection of statistical tests that cover data related to repeated measures. Table 3 summarizes the statistical approaches used in this study, the complete description of which can be found in [14], [28].

6.2 Analysis of the Results

This section discusses the results of the statistical tests conducted in order to evaluate hypotheses $H_{coup1.1}-H_{coup1.6}$ and $H_{coup2.1}-H_{coup2.6}$ defined in Section 6.1.3.

6.2.1 Descriptive Statistics

The descriptive statistics are presented in Tables 4, 5, 6 individually for each dependent variable, with the exception of the stability metric, Modification Impact Localization (MIL). The values of MIL are not included because all participants performed TASK-COUP2 without introducing any new system faults, and as such, all MIL values were equal to zero. Therefore, it is evident prima facie that there is no significant difference between the mean values of MIL obtained for the investigated lowly and highly-coupled elements, resulting in rejection of the alternative hypothesis $H_{coup1.4}$. The absence of failures is believed to be due to the small size of the services, and because the description of the tasks designed to evaluate the stability of a system (refer to [37]) instructed participants to make sure that any changes to existing functionality did not result in a negative effect on the system.

The values for the remaining two dependent variables, Failure Analysis Efficiency (FAE) metric (Table 4), and Modification Complexity (MC) metric (Tables 5 and 6), are shown for each of the lowly (Element1) and highly (Element2) coupled elements in each experimental service, where a lower value of a given dependent variable indicates better maintainability. Note that in contrast to FAE, which is only related to outgoing coupling (refer to Section 5), the MC metric is used to evaluate both incoming and outgoing coupling types given that: 1) The *changeability of a system* was hypothesized to be related to the number of incoming relationships (Hypothesis $H_{coup1.2}$), and 2) the *changeability of an element* was hypothesized to be related to the number of outgoing relationships (Hypothesis $H_{coup1.4}$).

7. Conventionally, a test with a power value of 0.8 (that is, $\beta < = 0.2$) is considered statistically powerful [14].

TABLE 3
Summary of the Statistical Analysis Techniques

GOAL-COUP1 ($H_{coup1.1}$ - $H_{coup1.6}$)
- A standard *paired t-test for dependent samples* was used to evaluate the relationship between the maintainability values collected for lowly- and highly-coupled elements in each experimental sub-system. More specifically, t-test allowed testing the null hypothesis that the means of a group of dependent variables collected for lowly-coupled elements are equal to a group of dependent variables collected for the highly-coupled elements. Note that the decision was made to use a paired t-test due to the nature of this empirical study in which repeated tests were conducted with two (paired) levels of treatment (i.e. two groups of maintainability values for low and high levels of coupling).   - The *Wilcoxon matched pairs* test was used as a non-parametric alternative to the paired t-test.
GOAL-COUP2 ($H_{coup2.1}$ - $H_{coup2.6}$)
- A *one-way ANOVA for repeated measures* was used to evaluate the relationships between the different types of coupling and the maintainability values collected for all highly-coupled elements in each experimental sub-system. The one-way ANOVA has the same theoretical foundation and purpose as the paired t-test, except that it is used to test for differences among at least three groups. The null hypothesis is that all the groups have the same mean, and the alternate hypothesis is that at least one of the means is different from the others.   - The *Kruskall-Wallis* test was used as a non-parametric alternative to one-way ANOVA approach.   - The *Fisher's Least-Significant Difference (LSD)* mean comparison test was applied for the purpose of comparing differences between group means in a pair-wise manner as part of one-way ANOVA.   NOTE: Due to the repeated nature of the tests, Bonferroni's adjustment procedure, in which the level of significance (α) of each individual test is adjusted downwards to reduce the overall risk of committing Type I error, was considered. The decision was made not to apply adjustment procedures because: i) they increase the chance of making a Type II error [30]; and ii) the LSD test was applied for the purpose of comparing differences between the individual groups of values, thereby already reducing the chance of committing a Type I error.

TABLE 4
Descriptive Statistics: TASK-COUP1
(FAE Values—Outgoing Coupling)

ELEMENT	*MIN*	*MAX*	*MEAN*	*MEDIAN*	*STD. DEV.*	*VAR.*
SER-COUP1b (covers **intra-service outgoing** coupling)						
Element1 *WISOCE = 1*	2	5	3.1	3	1.2	1.43
Element2 *WISOCE = 5*	4	10	6	5.5	1.76	3.11
SER-COUP2b (covers **indirect extra-service outgoing** coupling)						
Element1 *EESIOC = 1*	3	8	4.7	4	1.89	3.57
Element2 *EESIOC = 5*	5	12	7.9	7	2.56	6.54
SER-COUP3b (covers **direct extra-service outgoing** coupling)						
Element1 *WESOCE =1*	2	8	4.7	4	2.06	4.23
Element2 *WESOCE =5*	4	11	7.6	8	2.91	8.49

TABLE 5
Descriptive Statistics: TASK-COUP2
(MC Values—Incoming Coupling)

ELEMENT	*MIN*	*MAX*	*MEAN*	*MEDIAN*	*STD. DEV.*	*VAR.*
SER-COUP1a (covers **intra-service incoming** coupling)						
Element1 *WISICE =1*	2	10	5	4	2.4	5.78
Element2 *WISICE =5*	4	10	8.2	8.5	1.87	3.51
SER-COUP2a (covers **indirect extra-service incoming** coupling)						
Element1 *ESICSI =1*	5	9	6.3	5.5	1.57	2.46
Element2 *ESICSI =5*	8	15	11.2	11	2.2	4.84
SER-COUP3a (covers **direct extra-service incoming** coupling)						
Element1 *WESICE =1*	2	8	4.6	4.5	2.01	4.04
Element2 *WESICE =5*	6	16	10.8	11.5	3.65	13.3

Three general observations can be made by examining data shown in all three tables:

1. The means of FAE and MC (both incoming and outgoing) values collected for all lowly-coupled elements are smaller (i.e., better) compared to the means collected for the corresponding highly-coupled elements. This can be visualized in the graphs shown in Figs. 2, 3, 4, and will be tested for statistical significance in Section 6.2.2.
2. The means of FAE and MC (both incoming and outgoing) values collected for the elements exhibiting high indirect and direct extra service coupling are larger (i.e., worse) than the means collected for highly-coupled intraservice elements, but there is little difference between the means of elements exhibiting high indirect and direct extraservice coupling. This is shown graphically in Figs. 2, 3, 4, and will be tested for statistical significance in Section 6.2.3.
3. There is no obvious pattern in the dispersion (i.e., standard deviation [std. dev.] and variance [var.]) values for the examined lowly and highly-coupled

TABLE 6
Descriptive Statistics: TASK-COUP2
(MC Values—Outgoing Coupling)

ELEMENT	MIN	MAX	MEAN	MEDIAN	STD. DEV.	VAR.
SER-COUP1b (covers **intra-service outgoing** coupling)						
Element1 *WISOCE = 1*	4	8	5.6	5.5	1.51	2.27
Element2 *WISOCE = 5*	6	15	8.8	8.5	2.49	6.18
SER-COUP2b (covers **indirect extra-service outgoing** coupling)						
Element1 *EESIOC = 1*	3	8	5.2	5	1.4	1.96
Element2 *EESIOC = 5*	7	20	12.2	12	3.77	14.2
SER-COUP3b (covers **direct extra-service outgoing** coupling)						
Element1 *WESOCE =1*	2	10	5.1	5	2.23	4.99
Element2 *WESOCE =5*	5	20	12.4	12.5	3.89	15.1

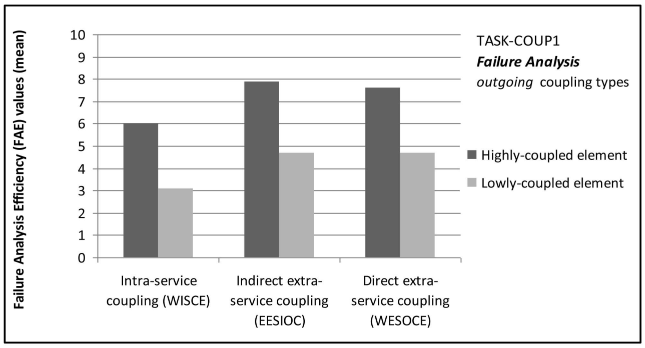

Fig. 2. TASK-COUP1—Failure Analysis Efficiency (FAE).

elements. Although the standard deviation and variance of FAE and MC (both incoming and outgoing) values collected for the highly-coupled elements are generally larger than those collected for the corresponding lowly-coupled elements; this can be attributed to the proportionally larger mean values. Specifically, the difference in *coefficient of variation (CV)*[8] values within all examined element pairs was low (< 0.1).

6.2.2 Hypothesis Testing: Maintainability Impact ($H_{coup1.1} - H_{coup1.6}$)

Nine *two-tailed paired t-tests for dependent samples* were conducted in order to examine the impact of low and high design-level coupling on the analyzability and changeability of service-oriented software according to hypotheses $\mathrm{H}_{\mathrm{coup1.1}} - \mathrm{H}_{\mathrm{coup1.6}}$, defined in Section 6.1.3. In summary, the results indicate that there is a statistically significant relationship between the SO coupling metrics presented in Section 5, and the ISO/IEC *analyzability* (FAE), and *changeability* (MC) metrics. This, in turn, suggests that all evaluated *coupling metrics can be used as early indicators of analyzability and changeability of service-oriented software.*

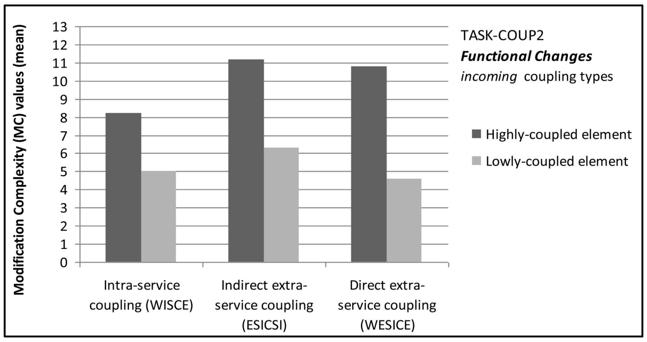

Fig. 3. TASK-COUP2—Modification Complexity (MC)—incoming.

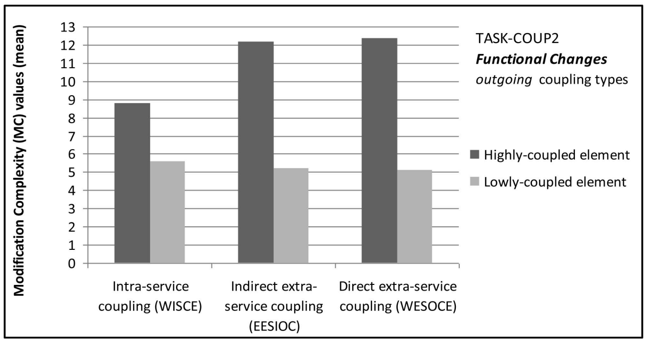

Fig. 4. TASK-COUP2—Modification Complexity (MC)—outgoing.

TABLE 7
Paired T-Test and Wilcoxon Matched Pairs Test Results for Hypotheses $\mathrm{H}_{\mathrm{coup1.1}} - \mathrm{H}_{\mathrm{coup1.6}}$ ($\mathrm{N} = 10$)

	$H_{coup1.1}$	$H_{coup1.2}$	$H_{coup1.3}$	$H_{coup1.4}$	$H_{coup1.5}$	$H_{coup1.6}$
Service	*SER-COUP1a*	*SER-COUP1b*	*SER-COUP2a*	*SER-COUP2b*	*SER-COUP3a*	*SER-COUP3b*
Test	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 WISICE=1 e2 WISICE=5	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 WISOCE=1 e2 WISOCE=5	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 ESICSI=1 e2 ESICSI=5	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 EESIOC=1 e2 EESIOC=5	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 WESICE=1 e2 WESICE=5	$\mu(e1_{DepVar})!= \mu(e2_{DepVar})$ e1 WESOCE=1 e2 WESOCE=5
Paired t-test (two-tailed p-values)						
DepVar =FAE	not tested	p = <0.0001	not tested	p = 0.0009	not tested	p = 0.012
DepVar =MC	p = 0.0003	p = 0.0046	p = 0.0004	p = <0.0001	p = <0.0001	p = 0.0001
Wilcoxon matched pairs test (two-tailed P-values)						
DepVar =FAE	not tested	P = 0.002	not tested	P = 0.0039	not tested	P = 0.0195
DepVar =MC	P = 0.0039	P = 0.002	P = 0.002	P = 0.002	P = 0.002	P = 0.002

Specifically, Table 7 shows the results of a two-tailed paired t-test used to determine whether the population means (μ) of the groups of maintainability values sampled for each lowly (e1) and highly (e2) coupled element in each of the experimental services are not equal (that is, $\mu(\mathrm{e1}_{\mathrm{Dep\ Var}})$!= $\mu(\mathrm{e2}_{\mathrm{Dep\ Var}})$, where DepVar stands for a dependent variable).

For example, the t-test related to Hypothesis $\mathrm{H}_{\mathrm{coup1.1}}$ (Table 7, column 2) was used to test inequality of the population means of the MC[9] values obtained for two experimental elements (e1 which has WISICE value of 1,

8. The coefficient of variation (CV) is a normalized measure of dispersion of a probability distribution, being defined as the ratio of the standard deviation to the mean. CV is useful when comparing a dispersion of data sets with different means [28].

9. The analyzability (FAE metric) was not evaluated in this particular test since it was hypothesised in Sections 5 and 6.1.3 that incoming coupling will not affect the analyzability of an element.

and e2 which has WISICE value of 5) belonging to the SER-COUP1a service. This particular test indicated a significant variance in the MC values collected for elements e1 and e2 ($\text{p value} = 0.0003$). Similarly, all remaining t-tests showed a significant variance in the FAE and MC values in each pair of lowly and highly-coupled elements at a statistically significant level of 0.05.

In addition, a nonparametric Wilcoxon matched pairs test also indicates a significant variance ($p < 0.05$) in all examined groups of FAE and MC values, as shown in Table 7. Therefore, *we accept alternative experimental hypotheses* $H_{coup1.1} - H_{coup1.6}$ and conclude that a highly-coupled SO design element has a significantly negative impact on the analyzability and changeability compared to a lowly-coupled element.

Note that this outcome is not surprising since it was shown in previous paradigms that the structural property of coupling is correlated to the maintainability of software. Nevertheless, accepting hypotheses $\text{H}_{\text{coup1.1}}-\text{H}_{\text{coup1.6}}$ gives us confidence that the investigated SO coupling metrics are empirically valid (or useful) measures of coupling.

6.2.3 Hypotheses Testing: Strength of Coupling ($H_{coup2.1} - H_{coup2.6}$)

Three *one-way ANOVA for repeated measures* tests were conducted to determine whether the specific types of coupling relationships have a significantly different impact on analyzability and changeability according to hypotheses $\text{H}_{\text{coup2.1}} - \text{H}_{\text{coup2.6}}$. The stability subcharacteristic was again not tested, given that all values of the corresponding metric (MIL) were equal to zero. In summary, the results of one-way ANOVA tests indicate that:

1. *All extraservice coupling metrics (ESICSI, EESIOC, WESICE, and WESOCE) should be weighted higher than the intraservice metrics (WISICE and WISOCE) in order to reflect their relatively stronger impact on the changeability of SO software*. This in turn suggests that the investigated extraservice metrics can potentially be more accurate indicators of changeability of service-oriented software, compared to the widely used OO metric, Coupling Between Objects (CBO), which is considered to be structurally equivalent to the intraservice WISICE/WESOCE metrics.
2. There is no statistically significant variance in the analyzability metric (FAE) values collected for the three different types of coupling relationships: intraservice outgoing coupling, indirect extraservice outgoing coupling, and direct extraservice outgoing coupling. The observed effects were in the hypothesized direction, but were not significant enough to reliably establish that both extraservice (indirect and direct) coupling types have a greater effect on the analyzability of SO software than intraservice coupling.

The specifics of the statistical tests are given below:

- Six *individual* test conditions (one condition per experimental hypothesis[10]), given below, were established in order to test the inequality of variance in the groups of analyzability and changeability values collected for six elements under study (shown in Table 8):
    - Test condition TC1 $(\text{H}_{\text{coup2.1}}) : \mu(\text{e1}_{\text{MC}}) != \mu(\text{e3}_{\text{MC}})$
    - Test condition TC2 $(\text{H}_{\text{coup2.2}}) : \mu(\text{e2}_{\text{FAE}}) != \mu(\text{e4}_{\text{FAE}}) \wedge \mu(\text{e2}_{\text{MC}}) != \mu(\text{e4}_{\text{MC}})$
    - Test condition TC3 $(\text{H}_{\text{coup2.3}}) : \mu(\text{e1}_{\text{MC}}) != \mu(\text{e5}_{\text{MC}})$
    - Test condition TC4 $(\text{H}_{\text{coup2.4}}) : \mu(\text{e2}_{\text{FAE}}) != \mu(\text{e6}_{\text{FAE}}) \wedge \mu(\text{e2}_{\text{MC}}) != \mu(\text{e6}_{\text{MC}})$
    - Test condition TC5 $(\text{H}_{\text{coup2.5}}) : \mu(\text{e3}_{\text{MC}}) != \mu(\text{e5}_{\text{MC}})$
    - Test condition TC6 $(\text{H}_{\text{coup2.6}}) : \mu(\text{e4}_{\text{FAE}}) != \mu(\text{e6}_{\text{FAE}}) \wedge \mu(\text{e4}_{\text{MC}})! = \mu(\text{e6}_{\text{MC}})$
- The individual test conditions were then *aggregated* according to the direction of coupling communication and the associated dependent variables in order to minimize the number of required tests (thereby decreasing the likelihood of committing a Type I error [28]), and evaluated using three one-way ANOVA tests:
    - ***Test 1*** (combines test conditions TC1, TC3, and TC5): $\boldsymbol{\mu(\text{e1}_{\text{MC}}) != \mu(\text{e3}_{\text{MC}}) != \mu(\text{e5}_{\text{MC}})}$;
    - ***Test 2*** (covers TC2, TC4, and TC6): $\boldsymbol{\mu(\text{e2}_{\text{FAE}}) != \mu(\text{e4}_{\text{FAE}}) != \mu(\text{e6}_{\text{FAE}})}$;
    - ***Test 3*** (covers TC2, TC4, and TC6): $\boldsymbol{\mu(\text{e2}_{\text{MC}}) != \mu(\text{e4}_{\text{MC}}) != \mu(\text{e6}_{\text{MC}})}$.

10. Note that, as described in Section 6.1.3, some hypotheses (e.g., Hypothesis $\text{H}_{\text{coup2.2}}$) were defined in terms of multiple dependent variables in order to minimize the total number of hypotheses, thereby improving the readability of the manuscript.

TABLE 8
Experimental Service Implementation Elements

ELEMENT	*ENCAPSULATING SERVICE*	*METRIC / VALUE*
e1	SER-COUP1a	WISICE = 5
e2	SER-COUP1b	WISOCE = 5
e3	SER-COUP2a	ESICSI = 5
e4	SER-COUP2b	EESIOC = 5
e5	SER-COUP3a	WESICE = 5
e6	SER-COUP3b	WESOCE = 5

For example, the aim of Test 1 is to determine whether the population means (μ) of the groups of changeability (MC metric) values collected for three elements exhibiting high incoming: intraservice (e1), indirect extraservice (e3), and direct extraservice (e5) coupling are not equal.

ANOVA results are presented below individually for each aggregated test (Tests 1-3).

- ***Test*** $1 : \mu(e1_{MC}) != \mu(e3_{MC})! = \mu(e5_{MC})$

Table 9 shows that the population means of the three sampled groups of MC values collected for the elements exhibiting high *incoming* coupling were not equal, with F statistic (3.68) exceeding the F critical value (3.35) at the statistically significant level of 0.05 ($\text{p-value} = 0.038$). Moreover, the results of a nonparametric Kruskal-Wallis test also show a significant variance between the groups of MC values (the Kruskal-Wallis' statistic $(\text{H}) = 6.63$, with $\text{p} = 0.036$ (corrected for ties)).

TABLE 9
One-Way ANOVA Results for MC Values (TASK-COUP2)—Incoming Coupling ($N = 30$)

SOURCE OF VARIATION	SUM SQUARES	MEAN SQUARE	F STAT.	F CRIT.	P-VALUE
Groups (between)	53.1	26.5	**3.68**	**3.35**	**0.038**
Residual (within)	194.8	7.2			
Total	247.9				

TABLE 10
LSD Groups Comparison of One-Way ANOVA Results for MC Values—Incoming Coupling ($N = 30$)

LSD CONTRAST	DIFFERENCE	95% CI	SIGNIFICANT
μ (e1$_{MC}$) != μ (e3$_{MC}$)	-3	-5.5 to -0.5	**yes**
μ (e1$_{MC}$) != μ (e5$_{MC}$)	-2.6	-5.1 to -0.1	**yes**
μ (e3$_{MC}$) != μ (e5$_{MC}$)	0.4	-2.1 to 2.9	no

The statistically significant ANOVA results indicate that there is more variation between tested groups than would be expected by chance, but they did not identify which specific group pairs are significantly different from one another. Therefore, the Fisher's Least-Significant Difference (LSD) mean comparison test was applied for the purpose of comparing differences between the individual groups of MC values. The results of this test are shown in Table 10, wherein a Boolean value reflects the significance of investigated group pairs. The results indicate a significant difference in population means of the MC values[11] for: 1) intraservice incoming coupling (e1) and indirect extraservice incoming coupling (e3), and 2) intraservice coupling (e1) and direct extraservice incoming coupling (e5); but the relationship between indirect (e3) and direct (e5) extraservice incoming coupling types is not significant at 0.05 level.

Therefore, *we accept the changeability-related component of the combined (refer to Section 6.1.3) alternative hypotheses $H_{coup2.1}$ and* $\mathrm{H}_{\mathrm{coup2.3}}$, but *reject the changeability-related component of the alternative hypothesis $H_{coup2.5}$*. This leads to a conclusion that both indirect and direct extraservice incoming coupling types have relatively stronger impact on the changeability of SO software compared to intraservice coupling.

- $$Test\ 2(H_{coup2.2}, H_{coup2.4},\ and\ H_{coup2.6}):$$
$$\mu(e2_{FAE})\ != \mu(e4_{FAE})! = \mu(e6_{FAE})$$

Table 11 shows that the F statistic (1.73) is below the F critical value (3.35) with p-value $= 0.197$. As such, we *reject the analyzability-related component of the alternative hypotheses $H_{coup2.2}$, $H_{coup2.4}$, and $H_{coup2.6}$* and conclude that different types of investigated coupling relationships do not result in a significantly different impact on the analyzability of service-oriented products. Consequently, the LSD mean comparison test was not conducted.

11. The LSD test does not indicate the relative direction of the groups' difference; therefore, the original mean values for each examined group (refer to Tables 4, 5, 6) should be used to interpret the direction of difference (e.g., $\mu(\mathrm{e1_{MC}}) < \mu(\mathrm{e3_{MC}})$).

TABLE 11
One-Way ANOVA Results for FAE Values (TASK-COUP1)—Outgoing Coupling ($N = 30$)

SOURCE OF VARIATION	SUM SQUARES	MEAN SQUARE	F STAT.	F CRIT.	P-VALUE
Groups (between)	20.9	10.4	**1.73**	**3.35**	**0.197**
Residual (within)	163.3	6			
Total	184.2				

TABLE 12
One-Way ANOVA Results for MC Values (TASK-COUP2)—Outgoing Coupling ($N = 30$)

SOURCE OF VARIATION	SUM SQUARES	MEAN SQUARE	F STAT.	F CRIT.	P-VALUE
Groups (between)	81.9	40.9	**3.46**	**3.35**	**0.046**
Residual (within)	319.6	11.8			
Total	401.5				

TABLE 13
LSD Groups Comparison of One-Way ANOVA Results for MC Values—Outgoing Coupling ($N = 30$)

LSD CONTRAST	DIFFERENCE	95% CI	SIGNIFICANT
μ (e2$_{MC}$) != μ (e4$_{MC}$)	-3.4	-6.6 to -0.2	**yes**
μ (e2$_{MC}$) != μ (e6$_{MC}$)	-3.6	-6.8 to -0.4	**yes**
μ (e4$_{MC}$) != μ (e6$_{MC}$)	-0.2	-3.4 to 3	no

- $$Test\ 3(H_{coup2.2}, H_{coup2.4},\ and\ H_{coup2.6}):$$
$$\mu(e2_{MC})\ != \mu(e4_{MC})! = \mu(e6_{MC})$$

Table 12 shows that the population means of the three sampled groups of MC values collected for the elements exhibiting high *outgoing* coupling were not equal, with the F statistic (3.46) exceeding the F critical value (3.35) at the statistically significant level of 0.05. This is further supported by the Kruskal-Wallis tests ($\mathrm{H} = 7$, with $\mathrm{p} = 0.03$).

The Fisher's Least-Significant Difference (LSD) test was again applied with the results, shown in Table 13, being similar to the results obtained in Test 1. To this end, *we accept the changeability-related component of the alternative hypotheses $H_{coup2.2}$ and $H_{coup2.4}$*, and *reject the changeability-related component of the alternative hypothesis $H_{coup2.6}$*.

In summary, the following conclusions can be made based on the results of Tests 1-3:

1. *Indirect and direct incoming and outgoing extraservice* coupling types, as measured by the ESICSI, WESICE, EESIOC, and WESOCE metrics respectively, should be weighted higher than intraservice coupling (measured using the WISICE/WISOCE metrics) because they have a stronger influence on the *changeability* of SO software. Note that the actual weight values are not defined in this paper due to a lack of empirical data. The weights could be established and validated in future work as discussed further in Section 7.
2. The ESICSI, WESICE, EESIOC, and WESOCE metrics appear to be more accurate indicators of the *changeability* of SO systems compared to the CBO metric, which is considered to be syntactically and structurally similar to the intraservice WISICE/WISOCE metrics. This is because WISICE/WISOCE, and correspondingly CBO, were designed to quantify the general coupling between elements (OO classes in a case of CBO) disregarding the strength of specific types of coupling relationships, which have been shown in this paper to have varying impact on changeability.
3. The current statistical evidence is not strong enough to establish that: 1) extraservice coupling types have a stronger influence on the *analyzability* of SO software compared to intraservice coupling, and 2) *direct extr-service* coupling should be weighted higher than *indirect extraservice* coupling. Nevertheless, since the obtained results were in the hypothesized direction, it is possible that the results would have been statistically significant if a larger number of participants were used in the study thereby making the study more sensitive to a smaller effect sizes (refer to Section 6.1.9).

6.3 Practical Implications

The experimental results presented in the previous section suggest that intraservice coupling can potentially have a lesser effect on maintainability than indirect and direct extraservice coupling. This indicates that it might be beneficial to design SO systems in terms of coarse-grained services so as to increase intraservice coupling and reduce extraservice relationships. However, since coupling is not an independent structural property, designers should also consider the *cohesiveness* of individual services [38]. Specifically, coupling and cohesion are commonly considered to be conflicting factors, and developing excessively coarse-grained services (in order to minimize extraservice coupling) can have a negative effect on their cohesion. To this end, one of the major challenges of SO design is to find the "right" balance between system-level coupling and service-level cohesion.

In terms of the general impact of low versus high coupling on maintainability, the experimental results were in line with expectations (and a general understanding of the structural property of coupling) showing significant differences in terms of the amount of effort required to correct and perfect SO software. Specifically, the results suggest that producing quality design artifacts with low levels of coupling can potentially reduce the time required to perform future maintenance tasks by 40-60 percent. Although additional project resources can be required to produce quality designs in the early stages of the SLDC, the maintainability improvements should leverage initial expenditure and decrease the total lifetime cost of the system. Moreover, it is expected that improvements in maintainability will be more pronounced in industrial scale systems containing larger services requiring more complex maintenance tasks.

6.4 Threats to Validity

This section describes threats to validity, primarily in terms of how they were alleviated, but also in terms of limitations in the few cases in which they were not adequately addressed. The threats are described according to the classification of Wohlin et al. [45].

6.4.1 Construct Validity

Construct validity refers to the degree to which experimental variables accurately measure the concepts they purport to measure.

The service-oriented coupling metrics used as the *independent variables* in the study were defined in a formal manner and also validated theoretically [35], thereby satisfying the criteria for construct validity. The *dependent variables* used to measure the subcharacteristic of maintainability are part of a standard suite of ISO/IEE metrics, and thus can also be considered constructively valid.

6.4.2 Internal Validity

Internal validity refers to the degree to which conclusions can be drawn about the causal effect of independent variables on dependent variables.

Differences among participants. The study used a within-subjects design; therefore, error variance due to differences among participants was reduced. Furthermore, purposive sampling [41] was employed to select participants with comparable knowledge and experience. Nevertheless, the development experience of participants was relatively broad, ranging from six months to five years, and as such, it would be desirable to conduct experimental studies targeting participants with more constrained experience profiles as described in Section 7.

Practice [learning and fatigue] effects. The random starting order with rotation [41] technique was used to assign participants to experimental services in a systematic counterbalancing manner in order to reduce any potential learning and fatigue effects.

Instrumentation effects. The experimental services were related to the same universe of discourse (AMS). Also, an effort was made to manipulate only the coupling of services, while keeping other structural properties (controlled variables) as constant as possible. Finally, a pilot study was conducted to ensure that the experimental tasks were comparable in terms of their complexity.

Anticipation effects. The participants were not told about the expected outcomes or the structural specifics of experimental services in order to ensure that expectations about specific levels of treatment did not influence their responses.

6.4.3 External Validity

External validity refers to the degree to which the results can be generalized to applied software engineering practice.

Participants. Experienced OO developers were recruited to participate in the study; therefore, our sample is likely to be representative of the overall population of software practitioners. However, the participants had more limited exposure to SOC. Consequently, we acknowledge that different results could be obtained with participants having more substantial experience with SO development.

Environment. A widely used IDE (Eclipse 3.3), modeling language (UML), and programming language (J2EE) were used in the experiment. These technologies can be considered as representative of an industrial environment.

Experimental materials and tasks. This particular threat was difficult to address due to the controlled nature of the experiment. Specifically, the experimental system may not be representative of an industrial product in terms of size, and the complexity of the maintenance tasks was low. Also, the participants were asked not to test the implemented changes, thereby posing a threat to participant effort since one can always implement a quick change poorly if one is not concerned with quality. Nevertheless, despite some limitations, we believe the findings of this study are valuable and provide a solid foundation for further empirical studies of the structural properties of SO software.

7 Conclusion and Further Research

This paper presented the results of a controlled experiment examining the relationship between SO design coupling, as measured by the metrics proposed by Perepletchikov et al. [35], and three specific subcharacteristics of software maintainability: *analyzability*, *changeability*, and *stability*. This was done to investigate: 1) the use of metrics as early indicators of the maintainability of service-oriented software, and 2) the relative strength of different SO coupling relationship types (*intraservice*, *indirect extraservice*, and *direct extraservice coupling)*.

The results of the experiment provide empirical evidence of the causal relationship between the investigated SO coupling metrics, and the dependent variables used in the study (ISO/IEC maintainability metrics). Specifically, the results suggest that highly-coupled elements have a negative influence on the *analyzability* and *changeability* of SO software compared to lowly-coupled elements disregarding of the specific coupling type or locus of impact.

As for the evaluation of the relative strength of coupling types, it was discovered that:

- Indirect and direct extraservice coupling relationships have a more negative effect on the *changeability* of SO software than intraservice relationships.
- The relative impact of indirect and direct extra and intraservice relationships on *analyzability* did not indicate any statistically significant differences.
- There was no statistical significance between any combination of incoming and outgoing indirect and direct extraservice coupling relationships. Nevertheless, it is expected that performing structural changes (e.g., merging or removing services) could show larger differences since, intuitively, it would be harder to perform such changes on a system containing elements coupled via direct extraservice relationships. Additionally, the reusability of such services could also be decreased.

While the size of the system under study was relatively small compared to industrial systems and the number of participants low, the experiment was carefully controlled, thereby allowing for a stronger case to be made for causation between the investigated variables. Nevertheless, this experiment should be replicated and extended in future work in order to confirm the validity of the initial results. Specifically, a complete family of experiments that captures multiple studies can be defined according to the framework of Ciolkowski et al. [13] along the following four dimensions:

1. Nature of the study and experimental tasks:
   a. Additional *controlled studies* should be conducted in order to *replicate* the overall experimental design and tasks of the study presented in this paper.
   b. Larger-scale *industrial studies* should be conducted in order to increase the external validity of the results (refer to Section 6.4). Larger studies would also facilitate the derivation of ratio scale metrics weights to improve the fidelity of maintainability prediction.
   c. Since the study conducted in this research evaluated only functional changes, future work could investigate the impact of structural changes (such as merging/removing services) on the maintainability of SO software. The complexity of the experimental failures and changes should also be varied.
2. Technology:
   a. The experimental system developed for the study presented in this paper was implemented using Java EE 5 with the business logic and rules encapsulated in stateless Session Beans. It would be advisable to also use existing business process orchestration languages (such as WS-BPEL) to encapsulate compositional and business logic in dedicated business process.
   b. Different implementation paradigms/languages (e.g., scripting languages such as PHP and procedural languages such as C) could be used to implement the experimental services.
3. Target systems:
   a. For each investigated type of coupling, the number of relationships could be varied across a number of linearly increasing levels (as opposing to the arbitrary chosen [low/high] levels used in the present study). This in turn would allow further evaluating the predictive capability of the coupling metrics using linear regression analysis, as well as providing more statistical power to reevaluate the negative findings presented in this paper.

   b. Large-scale, operational software products should be investigated, with the data required to calculate the dependent variables (ISO/IEC metrics) extracted from maintenance-related documentation collected over a longer period of time (for example, one or more years).
4. Participants:
   a. Larger sample sizes should be used to increase internal validity.
   b. Participants possessing greater experience with SOC should be used since those involved in the present study had limited exposure to service-oriented development.
   c. Participants with more constrained experience profiles should be sought (for example, *novice* software engineers with experience ranging from 1 to 12 months, and *experts* with experience ranging from 10 to 20 years).

ACKNOWLEDGMENTS

This work is funded by the ARC (Australian Research Council), under Discovery scheme no. DP0988345.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications.* Springer-Verlag, 2004.

[2] Analyse-it Software Ltd "Analyse-it 3.0 for Microsoft Excel—Standard Edition," http://www.analyse-it.com/, 2009.

[3] A. Arsanjani et al., "SOMA: A Method for Developing Service-Oriented Solutions," *IBM Systems J.,* vol. 47, no. 3, pp. 377-396, 2008.

[4] J. Bansiya and C.G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Trans. Software Eng.,* vol. 28, no. 1, pp. 4-17, Jan. 2002.

[5] V. Basili, G. Caldiera, and H.D. Rombach, "The Goal Question Metric Approach," *Encyclopedia of Software Eng.,* pp. 528-532, John Wiley & Sons, 1994.

[6] L.C. Briand, S. Morasca, and V.R. Basili, "Property-Based Software Engineering Measurement," *IEEE Trans. Software Eng.,* vol. 22, no. 1, pp. 68-86, Jan. 1996.

[7] L.C. Briand, J. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Trans. Software Eng.,* vol. 25, no. 1, pp. 91-121, Jan./Feb. 1999.

[8] L.C. Briand, S. Morasca, and V.R. Basili, "Defining and Validating Measures for Object-Based High-Level Design," *IEEE Trans. Software Eng.,* vol. 25, no. 5, pp. 722-743, Sept./Oct. 1999.

[9] L.C. Briand, C. Bunse, and J.W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs," *IEEE Trans. Software Eng.,* vol. 27, no. 6, pp. 513-530, June 2001.

[10] S. Cant, D. Jeffery, and B. Henderson-Sellers, "A Conceptual Model of Cognitive Complexity of Elements of the Programming Process," *Information and Software Technology,* vol. 37, no. 7, pp. 351-362, 1995.

[11] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE Trans. Software Eng.,* vol. 20, no. 6, pp. 476-493, June 1994.

[12] S.R. Chidamber, D.P. Darcy, and C.F. Kemerer, "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Trans. Software Eng.,* vol. 24, no. 8, pp. 629-639, Aug. 1998.

[13] M. Ciolkowski, F. Shull, and S. Biffl, "A Family of Experiments to Investigate the Influence of Context on the Effect of Inspection Techniques," *Proc. Sixth Int'l Conf. Empirical Assessment in Software Eng.,* p. 48, 2002.

[14] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences,* second ed., Erlbaum, 1988.

[15] M. Dagpinar and J.H. Jahnke, "Predicting Maintainability with Object-Oriented Metrics—An Empirical Comparison," *Proc. 10th Working Conf. Reverse Eng.,* p. 155, 2003.

[16] R. Dijkman and M. Dumas, "Service-Oriented Design: A Multi-Viewpoint Approach," *Int'l J. Cooperative Information Systems,* vol. 13, no. 4, pp. 337-368, 2004.

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

[18] T. Erl, *SOA: Design Patterns.* Prentice Hall, 2009.

[19] F. Faul, E. Erdfelder, A. Lang, and A. Buchner, "G*Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences," *Behavior Research Methods,* vol. 39, no. 2, pp. 175-191, 2007.

[20] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach,* second ed. Course Technology, 1998.

[21] N.E. Fenton and M. Neil, "Software Metrics: Roadmap," *Future of Software Eng.,* A. Finkelstein, ed., ACM Press, 2000.

[22] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using Metrics to Predict OO Information Systems Maintainability," *Proc. 13th Int'l Conf. Advanced Information Systems Eng.,* pp. 388-401, 2001.

[23] M. Genero, L. Jimenez, and M. Piattini, "A Controlled Experiment for Validating Class Diagram Structural Complexity Metrics," *Proc. Eighth Int'l Conf. Object-Oriented Information Systems,* pp. 483-487, 2002.

[24] A. Guruge, *Web Services: Theory and Practice.* Digital Press, Elsevier Science, 2004.

[25] B. Henderson-Sellers, *Object-Oriented Metrics: Measures of Complexity.* Prentice Hall, 1996.

[26] ISO/IEC "ISO/IEC 9126-1:2001 Software Engineering: Product Quality—Quality Model," 2001.

[27] ISO/IEC/IEEE "ISO/IEC 14764:2006, IEEE Std 14764-2006: Software Engineering: Software Life Cycle Processes—Maintenance," 2006.

[28] D. Levine, P. Ramsey, and R. Smidt, *Applied Statistics for Engineers and Scientists.* Prentice Hall, 2001.

[29] W. Li and S. Henry, "Maintenance Metrics for the Object Oriented Paradigm," *Proc. First IEEE Int'l Software Metrics Symp.,* pp. 52-60, 1993.

[30] R. Marcus, E. Peritz, and K. Gabriel, "On Closed Testing Procedures with Special Reference to Ordered Analysis of Variance," *Biometrika,* vol. 63, pp. 655-660, 1976.

[31] R. Moen, T. Nolan, and L. Provost, *Improving Quality through Planned Experimentation.* McGraw-Hill, 1991.

[32] M. Papazoglou, *Web Services: Principles and Technology.* Prentice Hall, 2007.

[33] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *Computer,* vol. 40, no. 11, pp. 38-45, Nov. 2007.

[34] J. Pasley, "How BPEL and SOA Are Changing Web Services Development," *IEEE Internet Computing,* vol. 9, no. 3, pp. 60-67, May/June 2005.

[35] M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari, "Coupling Metrics for Predicting Maintainability in Service-Oriented Designs," *Proc. 18th Australian Conf. Software Eng.,* pp. 329-340, 2007.

[36] M. Perepletchikov, C. Ryan, K. Frampton, and H. Schmidt, "Formalising Service-Oriented Design," *J. Software,* vol. 3, no. 2, pp. 1-14, 2008.

[37] M. Perepletchikov and C. Ryan, "A Controlled Experiment for Evaluating the Impact of Coupling—Experimental Material/Data," http://www.cs.rmit.edu.au/~mikhailp/research/TSE, online appendix, 2009.

[38] M. Perepletchikov, C. Ryan, and Z. Tari, "The Impact of Service Cohesion on the Analysability of Service-Oriented Software," *IEEE Trans. Services Computing,* vol. 3, no. 2, pp. 89-103, Apr.-June 2010.

[39] R. Pressman, *Software Engineering: A Practitioner's Approach,* sixth ed. McGraw-Hill Int'l, 2005.

[40] N.F. Schneidewind, "Methodology for Validating Software Metrics," *IEEE Trans. Software Eng.,* vol. 18, no. 5, pp. 410-422, May 1992.

[41] J. Shaughnessy, E.B. Zechmeister, and J.S. Zechmeister, *Research Methods in Psychology,* seventh ed., McGraw-Hill Humanities Social, 2005.

[42] M.P. Singh and M.N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents.* John Wiley & Sons, 2005.

[43] W. Stevens, G. Myers, and L. Constantine, "Structured Design," *IBM Systems J.,* vol. 13, no. 2, pp. 115-139, 1974.

[44] W. Wan Kadir and P. Loucopoulos, "Relating Evolving Business Rules to Software Design," *J. Systems Architecture: The EUROMICRO J.*, vol. 50, no. 7, pp. 367-382, 2004.
[45] C. Wohlin et al., *Experimentation in Software Engineering—An Introduction.* Kluwer Academic Publishers, 2000.
[46] E. Yourdon and L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice Hall, 1979.
[47] O. Zimmermann, M. Tomlinson, and S. Peuser, *Perspectives on Web Services: Applying SOAP, WSDL, and UDDI to Real-World Projects.* Springer Professional Computing, 2003.
[48] H. Zuse, *A Framework of Software Measurement.* Walter de Gruyter, 1998.

Mikhail Perepletchikov received the first class BAppSc degree in computer science (hons) and the PhD degree in computer science from RMIT University, Melbourne, Australia, in 2004 and 2009, respectively. His PhD thesis addressed software quality in the context of SOA. He is currently a research fellow and sessional lecturer in the School of Computer Science and Information Technology, RMIT University. He has substantial industry experience as a software engineer. His research interests include software development methodologies and processes, requirements engineering approaches, and software quality metrics. He is a member of the IEEE, the IEEE Computer Science Society, and the Australian Computer Society (ACS).

Caspar Ryan received the BAppSc degree in computer science (hons) and the PhD degree in computer science from RMIT University, Melbourne, Australia, in 1996 and in 2002, respectively. His PhD thesis presented a novel approach for studying the behavior of different groups of software engineers in practice. He is currently a senior lecturer in the School of CS & IT at the RMIT University. His current software engineering research involves metrics and software development methodology for SOA as a joint CI on an Australian Research Council Discovery grant. Other research involves adaptive systems and middleware. He was a joint inventor on three provisional US and Australian patents as a project leader within the Australian Telecommunication Cooperative Research Centre.