A Maintainability Assessment Model for Service-Oriented Systems

Twittie Senivongse and Assawin Puapolthep

***Abstract*—Web service technology has been part of many software systems for quite some time as the technology to drive business processes and enable reuse of software functionality and system integration. Service-oriented systems, like other software systems, need to undergo regular maintenance and hence maintainability is one key desirable attribute of such systems. This paper presents a maintainability assessment model for determining whether a service-oriented system is maintainable. The model follows the Quality Model of Object-Oriented Design assessment method or QMOOD to define a hierarchy of service maintainability attributes based on ISO/IEC 9126, i.e., analyzability, changeability, stability, and testability. These quality attributes are accompanied by a set of metrics that measure the quality of the service design as well as the quality of development and operational practice of the organization that hosts the service-oriented system. The assessment model is applied to the case of a communication solutions provider in Thailand and the assessment results can point the organization to the areas where it can improve maintainability.**

***Index Terms*—maintainability, Web services, measurement, SOA**

I. Introduction

WEB service technology has been around for quite some time as the key enabling technology for automation of business processes, reuse of software functionality, and integration of software systems. A Web service is described by an interface definition or Web Service Description Language (WSDL) that specifies the offered operations, messages, data parameters to be exchanged, and how to use the service. Different Web services form a Service-Oriented Architecture (SOA) system in which the functionalities of the services are orchestrated to achieve business functions [1]. Like other software systems, a service-oriented system need to undergo regular maintenance either for fixing errors, preventing errors, adapting to new environment, or adding new functionality. Maintainability is hence an important quality attribute of a service-oriented system. Maintainability assessment is useful for an organization to determine how maintainable its service-oriented system is and which areas need to be improved so that future maintenance would be easier and faster.

Maintainability is one of the software quality characteristics in ISO/IEC 9126 software quality model [2]. It bears on the efforts needed to make modifications to the software and is broken down to four subcharacteristics: 1) Analyzability characterizes the ability to identify the root cause of a software failure, 2) Changeability characterizes the amount of efforts to change a system, 3) Stability characterizes the negative impact that may be caused by system changes, and 4) Testability characterizes the amount of efforts needed to verify or test a system change. This paper presents a maintainability assessment model to assess the degree of maintainability of a service-oriented system. We follow ISO/IEC 9126 and adopt the Quality Model of Object-Oriented Design (QMOOD) assessment methodology [3] to define a hierarchy of service maintainability attributes and a set of accompanying metrics. The metrics either measure at the design of the services in the system (i.e., measure direct quality) or measure at the development and operation environment of the organization that hosts the service-oriented system (i.e., measure indirect quality). Metrics for direct quality are taken from the SOA design metrics in [4] whereas metrics for indirect quality are defined based on a questionnaire on development and operation environment of the organization. In an experiment, we apply the model to assess maintainability of the service-oriented system of a communication solutions provider in Thailand.

Section II of the paper discusses background and related work. Section III proposes the maintainability assessment model for service-oriented systems together with related metrics. Section IV applies the model to the case study and the paper concludes in Section V.

II. Background and Related Work

The Quality Model for Object-Oriented Design or QMOOD [3] is one of the quality models for assessment of the design of object-oriented software. QMOOD presents a quality model as a hierarchy of the quality attribute and the characteristics of the software design which reflect such a quality attribute. The hierarchy comprises four levels, linking the abstract quality attribute at the top level down to the more concrete characteristics at lower levels. An example is depicted in Fig. 1 [4]. At level 1 is the quality attribute that is the target of assessment. At level 2 are the more tangible quality-carrying design properties that can be directly assessed by examining the design components. Each

Manuscript received June 29, 2015; revised July 18, 2015.

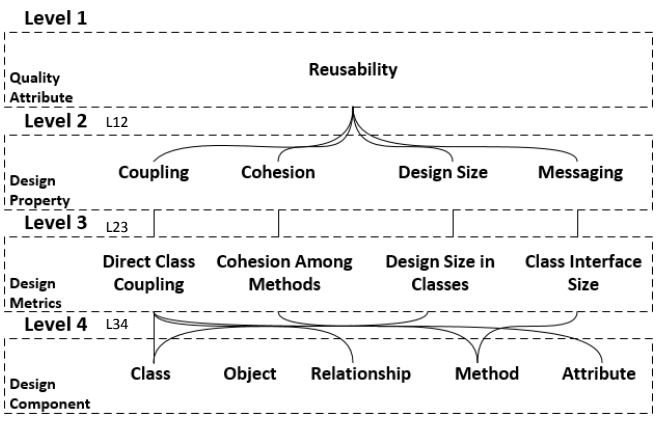
T. Senivongse is with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand (corresponding author phone: +66 2 2186996; fax: +66 2 2186955; e-mail: twittie.s@chula.ac.th).

A. Puapolthep was with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, and is now with True Corporation Public Company Limited, Bangkok 10310, Thailand (e-mail: assawin_pua@truecorp.co.th).

of the design property can be objectively assessed using one or more design metrics at level 3 where the targets for applying the design metrics are the design components at level 4. Linking the four levels are inter-level mappings L12, L23, and L34 that define relationships between levels. To define a quality model using QMOOD, a mathematical model is formulated based on a mapping between the quality attribute and the design properties that carry such quality (L12) as well as a mapping between each design property and the design metric that is used to assess that property (i.e., L23). Relative significance of individual design properties with positive or negative impact on the quality attribute can be weighted proportionately within the range of 0 to ± 1, and the summation of all weights for that quality attribute is either 1 or -1. The QMOOD method is simple to apply and the organization conducting a quality assessment can adjust the weights in the model as it sees fit.

Using the QMOOD method, Shim et al. [4] define a design quality model for service-oriented architecture. They define assessment models for five quality attributes of SOA design, i.e., effectiveness, understandability, flexibility, reusability, and discoverability. These quality attributes are influenced by a number of design properties, i.e., coupling, cohesion, complexity, design size, service granularity, parameter granularity, and consumability. They propose many service internal metrics, service external metrics, and system metrics that examine design components, i.e., services in the system, operations, calls, and messages. Their quality model does not address maintainability as they focus on attributes of service design artifacts, but a number of the proposed design properties and metrics will be adopted here.

On service maintainability, Zhe and Kerong [5] analyze the characteristics of service-oriented architecture and the factors affecting software maintainability, including quality of the software development method, developers, and document produced, as well as the degree of standardization and reuse. They propose a service-oriented software maintainability assessment method that takes into account the four subcharacteristics of ISO/IEC 9126 maintainability. The method addresses maintainability factors that should be considered but does not define any metrics. We will consider these factors when building our assessment model.



Reusability = -0.25*Coupling + 0.25*Cohesion + 0.5*Design Size + 0.5*Messaging

Fig. 1. An example of an OO software quality model built by QMOOD method.

Zarrin et al. [6] present a model to assess maintainability of SOA systems. The idea is quite similar to ours in that service structural properties in the design phase as well as service management mechanism structures in the operation phase are considered as effective factors in assessing service maintainability. In the design phase, the factors are three design properties from [4], i.e., coupling, cohesion, and granularity, and in the operation phase, the factors are the ITIL processes that are practiced in the system. However, their model is only conceptual, without specific assessment details. Other work focuses merely on structural properties of the design, such as the work by Perepletchikov [7] which focuses on cohesion and coupling as useful predictors of maintainability of service-oriented software. Leotta et al. [8] take a different approach and compare maintainability of non-SOA and SOA systems. They focus only on changeability at the architectural level, i.e., maintainability is determined by the number of architectural components of the system which are affected by the change requests as well as the level of efforts put to respond to the change requests.

III. Maintainability Assessment

Following QMOOD, we build a hierarchical maintainability model for service-oriented systems in Fig. 2.

Level 1 is the level of system quality attributes and is divided into three sublevels (see III.A). At level 1H is the high-level system quality attribute that is the target of assessment, i.e., maintainability. At level 1M are the medium-level system quality attributes, i.e., analyzability, changeability, stability, and testability. These attributes are further characterized by low-level system quality attributes at level 1L.

Level 2 is the level of service system properties. They are more tangible properties that show the quality attributes at level 1L (see III.B).

Level 3 is the level of service system metrics. They are used to objectively assess the service system properties at level 2 (see III.C).

Level 4 is the level of service system components, i.e., service, interface, message, program, business process, relationship between system components, and service environment. These system components are the targets of measurement by the service system metrics at level 3.

Mappings between the levels relate service system metrics and service system properties to system quality attributes, and form the assessment model (see III.D).

Fig. 2. Hierarchical maintainability model.

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. System Quality Attributes and Mapping to Service System Properties

As *maintainability* (level 1H) bears on the efforts needed to make modifications to the software, we consider a maintainable service-oriented system as one with the design and operation environment that are easy and fast to change and maintain. Based on ISO/IEC 9126, maintainability is characterized by analyzability, changeability, stability, and testability at level 1M, and these attributes are further characterized by system quality attributes at level 1L which are adapted from [5]. We mark the service system properties that carry the 1L attributes with [4] if they are adopted from [4], and with * if they are newly introduced here. (see the definitions of service system properties in III.B).

1) *Analyzability* (level 1M) characterizes the ability to identify the root cause of a software failure or the points to be fixed. Analysis of service code and operation environment is hence required. Analyzability is broken down into the following attributes:
   1.1) *Readability* (level 1L) is related to code convention or programming style that make the service program easy to read. An organization should follow standard naming convention and coding format.
   Service system property (level 2): *Readability level**
   1.2) *Understandability* (level 1L) characterizes the effort necessary to learn and comprehend the design. Dependency between services, complexity, size of the system/service/exchanged data, and the likelihood of the service being found by other services are the signs of understandability.
   Service system property (level 2): *Coupling, cohesion, complexity, system size, service granularity, parameter granularity,* and *consumability* [4].
   1.3) *Accessibility* (level 1L) characterizes the ability to access services in the system during the development and maintenance processes. The services should be accessible remotely, have interoperable interfaces, and be appropriately documented in order to aid the maintenance.
   Service system property (level 2): *Accessibility level**
2) *Changeability* (level 1M) characterizes the amount of efforts to change a system. It is broken down into the following attributes:
   2.1) *Coupling structure* (level 1L) is related to the strength of dependency between services in the system. The system with highly coupled services is more difficult to change.
   Service system property (level 2): *Coupling* [4]
   2.2) *Isolatability* (level 1L) is related to the strength of relationship between operations in a service. If the operations are cohesive, any change in the service and its operations is likely to be isolatable and other parts of the system would be less impacted.
   Service system property (level 2): *Cohesion* [4]
   2.3) *Functional coverage* (level 1L) is related to size or coverage of the function provided by a service as well as the amount of data exchanged with a service. Large size makes it likely that service change would be change to the detail within the service while operations and data are not affected, making it easier to change as other parts of the system are unaffected.
   Service system property (level 2): *Service granularity* and *parameter granularity* [4]
3) *Stability* (level 1M) characterizes the negative impact that may be caused by system changes. The system should be able to continue to service in the face of changes. Stability is broken down into the following attributes:
   3.1) *Availability* (level 1L) is related to the frequency and length of downtime when the services and system undergo maintenance and changes. Service and system redundancy and load balancing should be in place to provide continuous operation.
   Service system property (level 2): *Availability level**
   3.2) *Data encapsulation* (level 1L) is related to size of the data exchanged with a service. Exchange of the data of small size can generate lots of traffic between services which can make the system less stable.
   Service system property (level 2): *Parameter granularity* [4]
   3.3) *Independence of changes* (level 1L) is related to the effort to avoid changes that may impact many parts of the system. When change is made to a service, existing clients should be considered.
   Service system property (level 2): *Independence of changes level**
4) *Testability* (level 1M) characterizes the amount of efforts needed to verify or test a system change. Service and system tests should be done in an easy and efficient manner. Testability is broken down into the following attributes.
   4.1) *Environment focus* (level 1L) is related to the focus on service environment in operation when testing the system. Functional testing of individual services is not enough as testing should consider services in operation environment, their QoS, and how they operate with middleware and service bus as well as other external systems such as security, data, and legacy systems.
   Service system property (level 2): *Environment focus level**
   4.2) *Process simulation* (level 1L) is related to testing not only the services and service environment but also the process. Automated testing to simulate business process orchestration and input/output along the process flow is needed to test the behavior of the system.
   Service system property (level 2): *Process simulation level**

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B. Service System Properties and Mapping to Service System Metrics

The definitions of service system properties are as follows. We mark the service system metrics that are used to assess the service system properties with [4] if they are adopted from [4], and with * if they are defined here. (see the definitions of service system metrics in III.C).

1) *Readability level* (level 2) refers to the degree of readability of the services in the system.
   Service system metric (level 3): *Total readability**
2) *Coupling* (level 2) refers to the strength of dependency between services in the system.
   Service system metric (level 3): *Average number of directly connected services* [4]
3) *Cohesion* (level 2) refers to the strength of relationship between operations in a service.
   Service system metric (level 3): *Inverse of average number of used message* [4]
4) *Complexity* (level 2) refers to the difficulty of understanding relationship between services.
   Service system metric (level 3): *Number of operations* [4]
5) *System size* (level 2) refers to the size of the system.
   Service system metric (level 3): *Number of services* [4]
6) *Service granularity* (level 2) refers to the appropriateness of size of service.
   Service system metric (level 3): *Squared average number of operations to squared average number of messages* [4]
7) *Parameter granularity* (level 2) refers to the appropriateness of size of parameters.
   Service system metric (level 3): *Coarse-grained parameter ratio* [4]
8) *Consumability* (level 2) refers to the likelihood of other services to discover the given service.
   Service system metric (level 3): *Adequately named service and operation ratio* [4]
9) *Accessibility level* (level 2) refers to the degree of accessibility of the services in the system.
   Service system metric (level 3): *Total accessibility**
10) *Availability level* (level 2) refers to the degree of availability of the services in the system.
    Service system metric (level 3): *Total availability**
11) *Independence of changes level* (level 2) refers to the degree of independence of changes of the services in the system.
    Service system metric (level 3): *Total independence of changes**
12) *Environment focus level* (level 2) refers to the degree of environment focus on the system.
    Service system metric (level 3): *Total environment focus**
13) *Process simulation level* (level 2) refers to the degree of process simulation in the system.
    Service system metric (level 3): *Total process simulation**

C. Service System Metrics

Service system metrics from [4] measure at the design of the services in the system (i.e., measure direct quality) and are listed in Table I. Other metrics measure at the development and operation environment of the organization that hosts the service-oriented system (i.e., measure indirect quality) and we define them based on the questionnaire in Table II. The service team (i.e., team leader, analysts, developer, admin) answer each question by giving the score of 2 (Yes), 1 (Partly), 0 (No). Thus each metric calculates the total score for the questions under each service system property.

D. Maintainability Assessment Model

The maintainability assessment model for a service-oriented system is defined by equations in Table III. It follows the hierarchy of system quality attributes and service system properties. The equations are defined by the QMOOD method by which the weight of each system quality attribute/service system property is in (0, 1]. First, we determine whether the quality attribute/system property has positive or negative impact on the quality attribute of interest. The weight of -0.5 or -1 is assigned to the quality attribute/system property with a negative impact, and 0.5 or 1 to those with positive impact. Then we adjust the weights proportionately so that the summation of all weights for the quality attribute of interest is 1 or -1. QMOOD lets the organization conducting a quality assessment adjust the weights in the model as it sees fit.

TABLE I
SERVICE SYSTEM METRICS

Service System Metric	Definition	
Average number of directly connected services	$\frac{NDPS + NDCS}{SSNS}$	*NDPS* = Number of directly connected producer services in the system *NDCS* = Number of directly connected consumer services in the system *SSNS* = System size in number of services *TMU* = Total number of messages used in the system *NSO* = Number of Synchronous operations in the system *NAO* = Number of Asynchronous operations in the system *NFPO* = Number of fine-grained parameter operations in the system *NINS* = Number of inadequately named services in the system *NINO* = Number of inadequately named operations in the system
Inverse of average number of used message	$\frac{SSNS}{TMU}$	
Number of operations	$NSO + NAO * 1.5$	
Number of services	$SSNS$	
Squared average number of operations to squared average number of messages	$\frac{\left(\frac{NAO + NSO}{SSNS}\right)^2}{\left(\frac{TMU}{SSNS}\right)^2}$	
Coarse-grained parameter ratio	$\frac{NSO + NAO - NFPO}{NSO + NAO}$	
Adequately named service and operation ratio	$\left(\frac{SSNS - NINS}{SSNS * 2}\right) + \left(\frac{NSO + NAO - NINO}{(NSO + NAO) * 2}\right)$	

TABLE II
SERVICE SYSTEM METRICS QUESTIONNAIRE

Service System Metric	Question	Score (2= Yes, 1 = Partly, 0 = No)
Total readability = Sum of all readability scores	1. Program development follows standard programming convention and style, e.g., for naming, coding format, comment. 2. Standard data formats are used, e.g. organization-wide data schema. 3. Proper and meaningful naming is applied, e.g. to services, operations, parameters, variables. 4. Condition statements must be clear. 5. Programs must be adequately described and commented. 6. Programs have modular structure or are decomposed into components. 7. Operations or methods must not be too long, e.g., around 10-20 lines. 8. Naming, logic, and process within programs must conform to common usage within the business domain.	
Total accessibility = Sum of all accessibility scores	1. Activities and steps of the system process are documented. 2. Changes to systems and programs are documented. 3. The services and system can be accessed remotely for changes and maintenance.	
Total availability = Sum of all availability scores	1. Redundancy technique is employed to increase uptime by having more than one system component in operation at the same time for failover in the case of service and system failure. 2. Load balancing technique is employed to distribute loads from service invocation to different servers. 3. Frequency of downtime is appropriate. Service and system maintenance is planned and notified to service users in advance. 4. When a service is down, it takes short time to recover, e.g., within 15 minutes.	
Total independence of changes = Sum of all independence of changes scores	1. When a service is changed, the change is at the implementation code, not the interface. 2. Change is not likely to be made at a service that is used heavily by other services or other parts of the system. 3. For change that results in a new version of a service, the old and new versions are both kept in operation, i.e., to still accommodate old-versioned service clients. 4. Change that results in a new version of a service considers compatibility with the old version.	
Total environment focus = Sum of all environment focus scores	1. Service system testing is done on the infrastructure and middleware in the real operation environment. 2. When there are changes, the services and system are thoroughly tested. 3. Service availability is tested. 4. Service security is tested. 5. Service performance is tested. 6. Service interoperability with other systems is tested. 7. Automated tools are used in service and system testing.	
Total process simulation = Sum of all process simulation scores	1. Business process orchestration involving coordination of services must be tested. 2. Automated tools are used to simulate and test the business process.	

TABLE III
MAINTAINABILITY ASSESSMENT MODEL

System Quality Attribute	Model
Maintainability (1H)	$0.25 * Analyzability + 0.25 * Changeability + 0.25 * Stability + 0.25 * Testability$
Analyzability (1M)	$0.33 * Readability + 0.33 * Understandability + 0.33 * Accessibility$
Changeability (1M)	$-0.3 * Coupling\ structure + 0.65 * Isolatability + 0.65 * Functional\ coverage$
Stability (1M)	$0.33 * Availability + 0.33 * Data\ encapsulation + 0.33 * Independence\ of\ changes$
Testability (1M)	$0.5 * Environment\ focus + 0.5 * Process\ simulation$
Readability (1L)	*Readability level*
Understandability (1L) [4]	$-0.66 * Coupling + 0.25 * Cohesion - 0.66 * Complexity - 0.66 * System\ size + 0.25 * Service\ granularity + 0.25 * Parameter\ granularity + 0.25 * Consumability$
Accessibility (1L)	*Accessibility level*
Coupling structure (1L)	*Coupling*
Isolatability (1L)	*Cohesion*
Functional coverage (1L)	$0.5 * Service\ granularity + 0.5 * Parameter\ granularity$
Availability (1L)	*Availability level*
Data encapsulation (1L)	*Parameter granularity*
Independence of changes (1L)	*Independence of changes level*
Environment focus (1L)	*Environment focus level*
Process simulation (1L)	*Process simulation level*

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

As in Table III, the organization can put more weight on a certain quality attribute/system property if it is considered as having stronger impact on the quality attribute of interest. Note that, before applying the metric values to the model, the values have to be normalized in the range of [0, 1] as they are of different measurement units.

IV. Experiment

In an experiment, we use the maintainability assessment model to assess a Customer Service Management for Billing System of a communication solutions provider in Thailand. The first version of the system consists of 12 services and the second version has the functionality extended to 18 services. We use the service system metrics in Table I to measure the design of the services. For the questionnaire, we ask 6 system analysts and project leader with 4-to-7-year experiences to assess the two system versions, and record the average score for each question in order to calculate the average score of all questions with regard to each service system metric in Table II. The measured values and normalized values are in Tables IV and V. Normalization assumes the measured values of the first version as 1 and the values of the second version as the ratio to the first version.

TABLE IV
Service System Property Assessment Results

Service System Property	Assessed Value		Normalized Value	
	V. 1	V. 2	V. 1	V. 2
Coupling	1	1	1	1
Cohesion	0.08	0.05	1	0.63
Complexity	74	142	1	1.92
System size	12	18	1	1.5
Service granularity	0.22	0.17	1	0.77
Parameter granularity	1	1	1	1
Consumability	0.49	0.46	1	0.94
Readability level	1.48	1.56	1	1.05
Accessibility level	1.06	1.28	1	1.21
Availability level	1.33	1.46	1	1.1
Independence of changes level	1.46	1.54	1	1.05
Environment focus level	1.36	1.52	1	1.12
Process simulation level	0.75	0.83	1	1.11

TABLE V
System Quality Attribute Assessment Results

System Quality Attribute	Assessed Value		Normalized Value	
	V. 1	V. 2	V. 1	V. 2
Readability (1L)	1.48	1.56	1	1.05
Understandability (1L)	-0.98	-2.08	-1	-2.12
Accessibility (1L)	1.06	1.28	1	1.21
Coupling structure (1L)	1	1	1	1
Isolatability (1L)	0.08	0.05	1	0.63
Functional coverage (1L)	1	0.89	1	0.89
Availability (1L)	1.33	1.46	1	1.1
Data encapsulation (1L)	1	1	1	1
Independence of changes (1L)	1.46	1.54	1	1.05
Environment focus (1L)	1.36	1.52	1	1.12
Process simulation (1L)	0.75	0.83	1	1.11
Analyzability (1M)	0.99	1.45	1	1.46
Changeability (1M)	1	0.69	1	0.69
Stability (1M)	0.99	1.04	1	1.05
Testability (1M)	1	1.12	1	1.12
Maintainability (1H)	1	1.08	1	1.08

The assessment results conform to the expected quality changes during the redesign. The second version is a larger system with extended functionality, so it could be more complex and more difficult to understand and change. However, this communication solutions provider improves their service management in the second version of the system, so the overall maintainability is improved.

V. Conclusion

The proposed maintainability assessment model for service-oriented systems is based on assessment of both service design and service management aspects, and follows ISO/IEC 9126. The model can be used by organizations as a self-assessment tool to assess whether their service-oriented systems are maintainable, and which parts of the service design and which areas of service operation management should be improved. However, it is possible for an organization to adjust the model or extend it to take into account other system quality attributes. Also, the questionnaire can be extended to include more questions or follow a service management standard such as ITIL, but a trade-off has to be considered by including only necessary questions to keep the questionnaire practical to use. For future work, we plan to develop a tool to better automate the assessment by automatically analyzing service design and supporting the assessment by the service team. Statistical methods can be used to verify the correlation between system quality attributes and service system properties, and verify weight values in the model.

References

[1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005.

[2] ISO/IEC, *ISO/IEC9126-1: 2001 Software Engineering: Product Quality – Quality Model*, Geneva, 2001.

[3] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Trans. Software Engineering*, vol. 28, issue 1, pp. 4-17. Jan. 2002.

[4] B. Shim, S. Choue, S. Kim, and S. Park, " A design quality model for service-oriented architecture," in *Proc. 15th Asia-Pacific Software Engineering Conf.* (*APSEC 2008*), Beijing, China, 2008, pp. 403-410.

[5] M. Zhe and B. Kerong, "Research on maintainability evaluation of service-oriented software," in *Proc. 3rd IEEE Int. Conf. Computer Science and Information Technology* (*ICCSIT 2010*), Chengdu, China, 2010, pp. 510-512.

[6] M. Zarrin, M. A. Seyyedi, and M. Mohsenzaeh, "Designing a comprehensive model for evaluating SOA-based services maintainability," *Int. J. Computer Science and Information Security*, vol. 9, no. 10, pp. 51-57. Oct. 2011.

[7] M. Perepletchikov, "Software design metrics for predicting maintainability of service-oriented software," Doctoral Thesis, RMIT Univ., 2009.

[8] M. Leotta, F. Ricca, G. Reggio, and E. Astesiano, "Comparing the maintainability of two alternative architectures of a postal system: SOA vs. non-SOA," in *Proc. 15th European Conf. Software Maintenance and Reengineering* (*CSMR 2011*), Oldenburg, Germany, 2011, pp. 317-320.

ISBN: 978-988-19253-6-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)