Journal of Theoretical and Applied Information Technology
31st March 2015. Vol.73 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPARATIVE EVALUATION OF SERVICE INTERFACE APPROACHES TO SYSTEM INTEGRATION FOR OPERATION CENTER

[1]NORZIANA YAHYA, [2]ABDUL HANAN ABDULLAH
[1]Faculty of Computing, UTM, Skudai, Johor, Malaysia
[2]Faculty of Computing, UTM, Skudai, Johor, Malaysia
E-mail: [1]ziana76@gmail.com, [2]hananfsksm@gmail.com

ABSTRACT

System integration of an operation center has become more complex and heterogeneous due to rapid innovations in the IT industry. Problems arise when a new integration requirement emerges due to complexity of the system integration architecture. To simplify the system integration, interoperability has to be emphasized at the design stage. Architectural design is the stage wherein the process to identify the interfaces involved in the system integration needs to be considered. Approaches to simplifying system integration via service interface design have been the subject of many research studies. In this paper, case study on two (2) traffic operation centers in Malaysia i.e. *Transport Management Centre of Kuala Lumpur City Hall* and *Traffic Monitoring Centre of Malaysian Highway Authority* were conducted to identify gaps amongst the system integration approaches used in both operation centers. Three (3) prominent approaches of system integration used by both operation centers were chosen, explored and discussed, namely, method-oriented interface, message-oriented interface and resource-oriented interface. The result of a systematic comparison of the approaches mentioned is also presented. Six (6) criteria were established to make comparison on the approaches: *interoperability, uniformity, scalability, reusability, heterogeneity,* and *compatibility*. The objective was to determine the best contemporary approach to service interface design in centralized system integration. The outcome of the evaluation was examined and improvement to the service interface design is proposed. The study is very significant as an attempt to establish a practice reference for enhancement of the current system integration as well as serve as a guide for future deployment of operation centers.

Keywords: S*ervice Interface Design; System Integration; Method-Oriented Interface; Message-Oriented Interface; Resource-Oriented Interface*

1. INTRODUCTION

The complexity of systems integration increases in line with number systems involved in an operation center. This indirectly creates potential to present big problem to the operation center and make the system unmanageable. When a new integration requirement emerges, it can cause many conflicting interfaces. To prevent system integration from becoming too complex to manage, it is necessary to determine the components of the system integration as early as the design stage. Several studies have emphasized that the success of interactions among the systems is dependent on how well the service interfaces are exposed [6], [24].

An approach proposed by Wei et al. [26] was aimed at resolving the tight coupling problem and interface complexity of WebService based on an XML-RPC interactive model. Through the comparison and analysis of WebService based on REST and traditional XML-RPC, the REST-based WebService was proposed. The approach also explained the advantages of the interactive model based on REST in Web-scale applications and set out the design method of this WebService.

Zhao et al. [28] who studied the possible composition of the abstract resource and run time service methods had proposed a method for a RESTful Web service composition based on linear logic. The study had also introduced a formal definition of RESTful Web services which also covered the development of resource-oriented and self-declarative methods.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Many research efforts have dealt with the service interface in system integration. Zhang et al. [27] and Kalasapur et al. [9], the interface matching method was used in interface integration. The matching was done on the semantic description of the parameters' input and output. Another study stressed the importance of determining the architecture and design of the components, subsystems and processes, and the effect of the interface on the process of integration [20]. In this paper, three (3) prominent approaches related to service interface design, namely, the method-oriented, message-oriented and resource-oriented approaches are described, refers from [12] [8] as well as other related works [6] and [24].

In Section 2, each approach is briefly described. In Section 3, explanation of the criteria used in the evaluation of the approaches is provided. The evaluation results are presented in Section 4. Based on the results and analysis carried out, a solution model called Service Interface Mediator was recommended and described in section 5. Finally, the conclusion and future work is presented in Section 6.

2. APPROACHES IN SERVICE INTERFACE DESIGN

Poorly designed service interfaces give negative effect to all applications using them. In contrast, well-designed service interfaces can speed up integration development and make solution more responsive to business needs. As such service interface design approach play important roles in system integration [2], [16].

Many approaches to system integration with regard to service interface design have been identified by previous researchers [6] and [24]. Based on preliminary study conducted on the traffic operation centers mentioned earlier, 3 most common approaches to service interface design were chosen i.e. method-oriented, message-oriented and resource-oriented approaches. Each approach is described in the following sub-sections.

2.1 Method-Oriented Interface

Method-oriented interface is a design approach that allows a program or application to call procedures located in other domains or machines. The application is exposed as one or more network objects, each with a unique set of functions or service interfaces which can be invoked. The service interfaces have a large set of operations and each operation performs a certain function. Service consumers have to know the exact definition of the service interface. In an environment where separate applications are communicating, any changes to the interface will require the service to be updated. This type of design can cause tightly coupled interfaces and also cause a lot of work in a large system in the case of changes to the interfaces. A good example of a method-oriented interface is the *Remote Procedure Call* (RPC) [11].

2.2 Message-Oriented Interface

In message-oriented design, service consumers consume a service defined in message structures instead of invoking function calls. The service endpoint embedded in the messages is sent to the Web service. In message-oriented design, the interface is fixed and changes are only made to the message structure. All the messages are described by using XML schema. However, the design makes it difficult to interpret and understand the functionality provided by a service. The structure of the messages that the service can handle needs to be examined in order to understand the functionality of the service. In message-oriented design, XML plays a major role, whereas the Web Services Description Language (WSDL) plays a minor role.

2.3 Resource-Oriented Interface

The resource-oriented interface or constrained interface [18] is an interface that adheres to a fixed set of standardized operations. An example of a resource interface is HTTP. HTTP defines the operations of PUT, POST, GET, and DELETE which are then applied to resources, located with Unified Resource Locators (URLs). With constrained interfaces, it is possible to build large distributed systems. Since the interface is standardized, it does not have to be updated. REST-style architecture, resource-oriented interfaces and content-oriented interfaces are used in lieu of constrained interfaces.

3. COMPARATIVE EVALUATION CRITERIA

This section briefly describes the criteria applied for the evaluation of the three (3) approaches to service interface design. The capability of the resulting system is analyzed in relation to how well it meets some relevant criteria. The indicators used to measure the capability are ranked as low, medium and high. A brief explanation of each criterion in the context of this research is provided in the following sub-sections.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1 Interoperability

Interoperability is the ability of two or more systems to work together to allow for information exchange [23] to enable them to operate effectively together [7] by adhering to common standards. The integration of different systems to use different data models and formats is achieved through common communication languages and protocols.

3.2 Uniformity

Uniformity of interface refers to shared terminology or mechanism that promote shared data model for interacting abstract objects from different applications [13]. The main objective is to promote a single method or mechanism that applies to all the interfaces involved in the system integration.

3.3 Scalability

Scalability is the ability of a system, or application to handle a growing number of integrated components in a capable manner. It also demonstrates the ability of a system to be enlarged to accommodate its growth. The base concept of scalability is the ability for a system or application to accept increased components without impacting its performance and objective [1]. A system is scalable if its performance improves after adding new component, proportionally to the capacity added.

3.4 Reusability

Reusability of interface refers to the ability of an interface which has functionalities to be reused for composing a new service or interface. This encourages interfaces with extra capabilities to be built for future usage scenarios. The interfaces also could be reused by multiple business processes. Reusable functions/logics reduce implementation time and effort, increase quality of service, and localize code modifications when a change in implementation is required. Thus, it eliminates the need for creating a new interface entirely and also can be reused by various business processes instead of reused by a particular business process only.

3.5 Heterogeneity

Heterogeneity refers to heterogeneous interface through which various systems with different platforms are able to be integrated. The heterogeneous interface demonstrates higher degree of interoperability amongst systems containing data resources with multiple types of formats. Standard principles conforming software interfaces used in common by different systems allowing them to communicate with each other.

3.6 Compatibility

Compatibility is a characteristic of software or system components which can operate satisfactorily together. It refers to the visibility of the same function or interface to be used by two or more applications or systems which are intended to operate cooperatively on the same or on different computers. They may also be compatible in one environment and incompatible in another [10].

4. RESULTS OF APPLYING EVALUATION CRITERIA

This section presents the results of the comparative evaluation of each of the three approaches to system integration. Table 1 shows the grading scheme for all the criteria used in the comparative evaluation. Table 2 summarizes the results of the comparative evaluation.

Table 1: Grading Scheme For All Criteria Used In Comparative Evaluation.

Criteria	Grading
Interface Features	
Interoperability	Low to High
Uniformity	Low to High
Extensibility	Low to High
Scalability	Low to High
Reusability	Low to High
Compatibility	Low to High

4.1 Interoperability

4.1.1 Method-Oriented Interface

The method-oriented interface conforms to SOAP and XML specifications of which its implementation depends on. In the case of Action values of " " and null in the SOAP specifications, different implementations may interpret the values differently due to ambiguous definition in the specification. Not all implementations support both values and this may lead to non-interoperable. It is also not clear how an XML-RPC with a void return and no out parameters should be represented as a service. It could be represented as an empty envelope or as an empty response element or as "No Response" code (HTTP 204).

Conclusion: low interoperability

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.1.2 Message-Oriented Interface

The message-oriented interface able to accommodate changes of services of a business for its information systems. It able to send and receive messages between distributed systems over heterogeneous platforms. It also creates loose coupling between participants in the systems. Integration with heterogeneous components is formed by an interface layer that allows them to communicate despite their differences. In this way, applications distributed on different network nodes able to communicate among them regardless of operating environment that they are hosting. The application interface able to link the applications without adapting source and target systems.

Conclusion: high interoperability

4.1.3 Resource-Oriented Interface

Most resource-oriented approaches require an HTTP library to be available for most of the operations. Uniform Resource Identifier (URI) in HTTP is a string of characters used to identify a name of a resource. This is an important concept of resource as global identifier. An application interacts with a resource (e.g. document or image) by knowing the resource identifier and the action required. With HTTP, resources can be manipulated and their representations can be exchanged across different domains.

Conclusion: high interoperability

4.2 Uniformity

4.2.1 Method -Oriented Interface

The concept of uniformity is to promote generalization of interfaces between interacting components or applications in system integration. In method-oriented architecture, a client must know exactly the object identity and also the object type prior to communicate with an application. The application is exposed as one or more network objects, each with a unique set of functions or interfaces which can be invoked. This situation creates low uniformity due to limitation of generalization of the interfaces.

Conclusion: low uniformity

4.2.2 Message-Oriented Interface

The evaluation of message-oriented interface is same as for method-oriented interface. Both interfaces cannot be *generalized* into a single interface, as such we need to treat each kind of interface name differently. The result is also same as the method-oriented interface.

Conclusion: low uniformity

4.2.3 Resource-Oriented Interface

In resource-oriented architecture, all networked resources are defined and addressed in a standard way in which they share a uniform interface to transfer their respective states. This is one of the most distinctive features of the resource-oriented architecture due to specific set of constraints imposed on the behavior of interacting components. The set of constraints ensure that the interactions use a consistent interface [15].

Conclusion: high uniformity

4.3 Scalability

4.3.1 Method -Oriented Interface

Low interoperability and reusability in method oriented architecture limits its scalability. Please refer to interoperability and reusability sections pertaining to the method oriented interface.

Conclusion: low scalability

4.3.2 Message -Oriented Interface

The message-oriented interface provides the most scalable way for sharing data and functionality. It is suitable for integration in large transaction volumes. This is due to the nature of the messaging type interfaces which does not require the client to suspend its work until complete.

Conclusion: high scalability

4.3.3 Resource-Oriented Interface

The resource-oriented interface is more scalable and is more maintainable over time. Simplicity is the key factor due to the constraints of the design style. The interface is easier to understand and work with and more predictable. The client-server architecture in resource-oriented simplifies component implementation and increases the scalability of server components [3]. Due to its high reusability and uniformity, the composition of new components is not an issue and these factors encourage the growth of integrated systems.

Conclusion: high scalability

4.4 Reusability

4.4.1 Method -Oriented Interface

The method-oriented interface offers the separation of functionality between client and server components. Each component has to focus on a particular function. Reusability of the server functionality across the client components is limited within a certain domain and platform only.

Conclusion: low reusability

Journal of Theoretical and Applied Information Technology
31st March 2015. Vol.73 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.4.2 Message-Oriented Interface

The message-oriented approach provides more reusable functions compared to the method-oriented approach due to the sent or received messages being in text form (XML format). The modification of contents of the message does not affect the integration interoperability. However, the limitation of interface generalization may decrease the reusability of its functions.

Conclusion: medium reusability

4.4.3 Resource-Oriented Interface

Resource-oriented architecture demonstrates higher degrees of decoupling and interoperability between components. In addition, abstraction of its interface uniformity promotes decoupling and independence between interacting components, leading to evolution-tolerance and reusability [13].

Conclusion: high reusability

4.5 Heterogeneity

4.5.1 Method -Oriented Interface

The method-oriented interface is a client/server infrastructure. Even though it allows interaction over different machines but it still depends on the language, platform and protocol used for the integration. It is interdependent and requires the simultaneous availability of all subsystems.

Conclusion: low heterogeneity

4.5.2 Message-Oriented Interface

The message-oriented architecture supports interoperable systems and applications interactions over a network where clients and servers are platform-independent. It also enables heterogeneous systems communicate over the HTTP protocol used on the Web.

Conclusion: high heterogeneity

4.5.3 Resource-Oriented Interface

The resource-oriented interface is designed to support various interoperable systems interacts over network world-wide regardless of their platforms. As such, the architecture able to support high degree of heterogeneity and as of today it is the most widely implemented in the world.

Conclusion: high heterogeneity

4.6 Compatibility

4.6.1 Method -Oriented Interface

For low-scale and simple integration, the development efforts for the method-oriented interface are low but when the systems grow it may become complex. Changes in one system may have effects on a few systems and this may cause incompatibility.

Conclusion: low compatibility

4.6.2 Message-Oriented Interface

The message-oriented interface is a text-based environment which is common for all systems, and compatibility is therefore not an issue.

Conclusion: high compatibility

4.6.3 Resource-Oriented Interface

Due to its uniformity features, the resource-oriented interface will give high compatibility for all applications dealing with it within the same environment.

Conclusion: high compatibility

As described in this section, each interface design approach has advantages and disadvantages based on the selected criterion. The results presented in Table 2 show that the resource-oriented interface is the best solution for system integration.

Table 2: Result Of Comparative Analysis Of The Three System Integration Approaches.

Criteria	Method-Oriented	Message-Oriented	Resource-Oriented
Interface Features			
Interoperability	Low	High	High
Uniformity	Low	Low	High
Scalability	Low	High	High
Reusability	Low	Medium	High
Heterogeneity	Low	High	High
Compatibility	Low	High	High

5. SERVICE INTERFACE MEDIATOR

Based on the analysis outcome stated in previous section, a new interface concept called Service Interface Mediator (SIM) was proposed to provide a unified service interface as depicted in Figure 1.

Journal of Theoretical and Applied Information Technology
31st March 2015. Vol.73 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

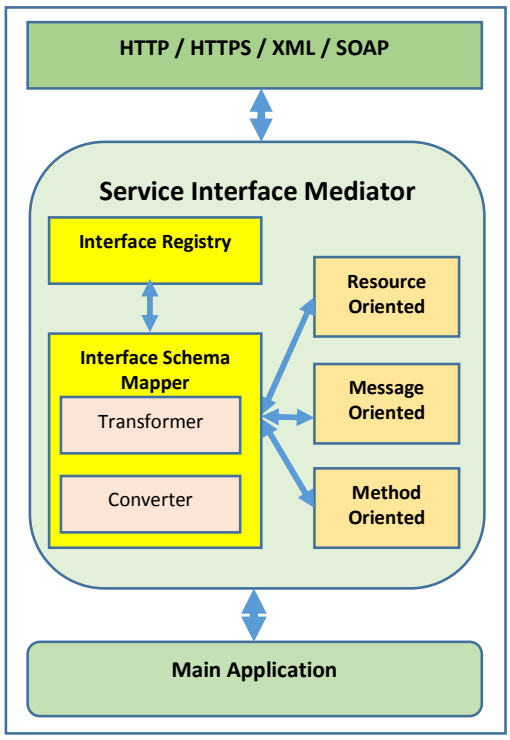

Figure 1: Preliminary Architecture of Service Interface Mediator

The SIM comprises of five main components as described below:-

a. Interface Schema Mapper (ISM)
Service Interface Mediator is the first point contact between any integrated system and main application. The first component to identify the service interface type from service consumer is Interface Schema Mapper (ISM). The ISM has two (2) subcomponents i.e. Transformer and Converter. The Transformer is used to transform service(s) into compatible form agreed between integrated systems. The Converter such as XML converter is used to do data conversion in order to make it compatible and acceptable to the receiving system.

b. Interface Registry (IR)
IR is where all service interfaces to be published are registered.

c. Method-oriented Interface(MT)
Service Interface will invoke an application-specific operation based on the service name given.

d. Message-oriented Interface(MS)
Service Interface will read the body of message and do appropriate action based on the instruction given.

e. Resource-oriented Interface(RS)
Service Interface will request the target resource based on the input given.

In the SIM application, the client browser or applications invokes an application-specific operation on a service endpoint with input arguments. WSDL files will be used by the main application to describe the operations that the interface supports and the parameters that the operations handle. XML schema will be used to describe the interface schema parameter structures.

6. CONCLUSION AND FUTURE WORK

The most important factor in establishing system integration is interoperability. Depending on the operational requirements and constraints of an operation center, the selection of the approach is crucial. The decision to use a message-oriented or method-oriented approach depends on the choice of protocols, architecture and products.

The resource-oriented interface was found to be the best approach in this study. It applies Web principles to design which make a system easy to maintain. Web services use a uniform set of operations, so it stands to reason that there is less complexity and high compatibility in the resource-oriented interface. Due to its uniform features, such an interface is able to reduce the cost by ensuring it is only written once, rather than once for each application it has to deal with.

Since the method-oriented approach works on object interfaces, changes in one system may affect another system interacting with it. The system complexity will become high when the system expands and much effort needs to be put in to maintain the system; this increases the total cost ownership. The message-oriented approach is able to reduce the system complexity. As such, it provides high compatibility.

Selection of a single approach to systems integration for an operations center is not very practical due to various systems with various technologies involved in the system integration. Therefore, the concept of Service Interface

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Mediator is an option for the integration solutions. Further study on technological support within the framework of the Service Interface Mediator is required. Study on limitations and advantages of present technology will help to generate a better interface design. The design shall serves as a reference and guide for future operation centers.

7. ACKNOWLEDGEMENT

The author wishes to express their sincere gratitude to the *Transport Management Centre of Kuala Lumpur City Hall* and *Traffic Monitoring Centre of Malaysian Highway Authority* for providing very good platforms to pursue these research activities. For further information about both operation centers, please refer to *www.itis.com.my* and *www.llm.gov.my* respectively.

REFRENCES:

[1] Bondi, A.B. (2000). Characteristics of scalability and their impact on performance. Proceedings of the 2nd International Workshop on Software and Performance, Ottawa, Ontario, Canada, ISBN 1-58113-195-X, pp. 195 - 203.

[2] Djavanshir, G. R. and Khorramshahgol, R. (2007). Key Process Areas in Systems Integration. IT Professional, vol. 9, pp. 24-27.

[3] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Phd Thesis, University of California, Irvine.

[4] Garlan, D. (2000). Software Architecture: a Roadmap. ICSE '00 Proceedings of the Conference on The Future of Software Engineering.

[5] Gorton, I. (2006). Essential Software Architecture. Springer-Verlag.

[6] Henkel, M. and Zdrakovic, J. (2005). Approaches to Service Interface Design. In Proceedings of the Web Service Interoperability Workshop, First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'2005). Geneva, Switzerland: Hermes Science Publisher.

[7] Hura, M., McLeod, G., Schneider, J, et al. (2000). Interoperability: A continuing Challenge in Coalition Air Operations. Chapter 2 "A broad Definition of Interoperability". RAND Monograph Report, 2000.

[8] Integrated Transport Information System of Dewan Bandaraya Kuala Lumpur (DBKL). [Online]. Available: http://www.itis.com.my

[9] Kalasapur, S., M. Kumar, et al. (2005). Seamless service composition (SeSCo) in pervasive environments. Proceedings of the first ACM international workshop on Multimedia service composition. Hilton, Singapore, ACM: 11-20.

[10] Krishna, C. M., Shin, K. G. (1997). Real-Time Systems (McGraw-Hill Series in Computer Science).

[11] Krzyzanowski , P. (2014).Distributed Systems. Chapter 3: Remote Procedure Calls Remote Procedure Calls. [Online]. Available: http://www.cs.rutgers.edu/~pxk/416/notes/15-rpc.html

[12] Lebuhraya Malaysia (LLM). [Online]. Available: http://www.llm.gov.my

[13] Marvin V. Zelkowitz (1989). Requirements for a Software ccsEngineering Environment: Proceedings of the University of Maryland Workshop, May 5-8, 1986.

[14] Michael, A., K. Kostas, et al. (2011). Towards an Interpretation Framework for Assessing Interface Uniformity in REST. Proceedings of the Second International Workshop on RESTful Design: 47-50.

[15] Mitchell, K. (2011) A Matter of Style: Web Services Architectural Patterns. Chief Architect, Agogo Networks, Inc.Vienna Virginia, USA.

[16] Nilsson, E.G, Nordhagen, E. K, Oftedal. G, (1990). Aspects of System Integration, Center for Industrial research (SI).

[17] O'Connor, J.T, Hubers, M.J. (2002). System Integration Standards Development Efforts For Facilities Engineering and Construction. A Report for Construction Industry Study. University of Texas at Austin.

[18] Orchard, D., (2003). The four Major Constraints to Loosely Coupled Web Services, Webservices.org, [Online]. Available: htttp:///www.webservices.org/index.php/article/articleprint/1246/1/24/

[19] Peng, Y.-Y., S.-P. Ma, et al. (2009). REST2SOAP: A framework to integrate SOAP services and RESTful services. IEEE International Conference on Service-Oriented Computing and Applications, SOCA' 09, December 14, 2009 - December 15, 2009, Taipei, Taiwan, IEEE Computer Society.

[20] Rasmi, J., Anithashree, C., et al. (2008).Exploring the Impact of Systems Architecture and Systems Requirements on Systems Integration Complexity. IEEE Systems Journal, vol. 2, no. 2, June 2008. IEEE Computer Society.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[21] Rowe, D., Leaney, J.R (1997) Evaluating evolvability of computer based systems architectures - an ontological approach, Proc. International Conference on the Engineering of Computer Based Systems.

[22] Silva, A. C. and Loureiro. G., (2011). System integration issues - Causes, consequences & mitigations. Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on.

[23] Slater, T., (2012). "What is Interoperability?", Network Centric Operations Industry Consortium - NCOIC, 2012.

[24] Teo, H. M. and Kadir, W. M. N. W. (2006). A Comparative Study of Interface Design Approaches for Service-Oriented Software. XIII ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE (APSEC'06).

[25] Watson, R. (2011). RESTful Web Applications and Services. Burton IT research.

[26] Wei, N., M. Song, et al. (2011). A novel WebService architecture based on REST. International Conference on Smart Materials and Intelligent Systems, SMIS 2010, December 17, 2010 - December 20, 2010, Chongqing, China, Trans Tech Publications.

[27] Zhang, J., Yu, S., Ge, X., Wu, G. (2006). Automatic Web Service Composition Based on Service Interface Description. ; In International Conference on Internet Computing (2006):120-126

[28] Zhao, X., E. Liu, et al. (2011). A two-stage RESTful web service composition method based on linear logic. 9th European Conference on Web Services, ECOWS 2011. September 14, 2011 - September 16, 2011, Lugano, Switzerland, IEEE Computer Society.