Journal of Computing Research and Innovation (JCRINN) Vol. 6 No. 1 (2021) (pp44-53)
https://jcrinn.com : eISSN: 2600-8793 / DOI: 10.24191/jcrinn.v6i1.148

An Overview of Service Interface Design Approaches for Interoperability of Traditional System Integration Patterns

Norziana Yahya[1]*, Mohd Azahani Md Taib[2]
[1] Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA, Perlis Branch, Arau Campus, 02600 Arau, Perlis
[2]RichTech Synergy Sdn Bhd, Menara Maxis, 26th Floor, KLCC, Kuala Lumpur, Malaysia

*Corresponding author: *norzianayahya@uitm.edu.my*
Received Date: 31 August 2020
Accepted Date: 16 January 2021
Published Date: 14 March 2021

HIGHLIGHTS

- Interoperability issues in legacy systems which use traditional System Integration Patterns (SIP).
- Concept review on traditional SIP.
- Service Oriented Architecture with Web Services as a model for development of service interfaces architecture.
- Service Interface Design Approaches as solution for upgrading traditional SIP.

ABSTRACT

One of the major issues in system integration is to deal with interoperability of legacy systems which use traditional System Integration Patterns (SIP). Information are unable to exchange effectively when the systems involved comes from developer that tended to not interoperate and this leads to the interoperability problem in heterogeneous system integration. To address the interoperability issues, interfacing processes need to be made more easily by defining components, processes, and interfaces that affect the system integration architecture at the initial design stage. This paper includes a basic concept on types of traditional SIP covering File-Based, Common Database, Remote Procedure Call (RPC), Distributed Objects, and Messaging. An overview of three Service Interface Design (SID) approaches for systems interoperability is discussed. The discussions on these approaches serve as a basis for the solution of interoperability of heterogeneous systems which use traditional SIP.

Keywords: *Interoperability, Service Interface Design, Service Oriented Architecture, System Integration*

INTRODUCTION

Over the years, system integration (SI) has become more complex and more heterogeneous due to rapid innovation in the IT industry. The complexity increases when the number of systems involved increased. Systems need to share information by sharing data and functionality of systems involved but it comes from manufacturer or developer that tended to not interoperate (L'Amrani et al., 2020; Buyle, 2017; Roshen, 2009). Impacts from this situation, information is unable to exchange effectively and this leads to the interoperability problem in SI (Paniagua et al., 2019; Masethe et al., 2016). Things get worse when this legacy system uses traditional SI methods that are less interoperable. In this paper, interoperability refers to the ability of two or more systems to work together, to allow for information exchange (Jakimoski, 2016) as well as to enable them to operate effectively together by adhering to common standards. The integration

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

of different systems to use different data models and formats can be accomplished through mutual communication languages and protocols.

One way of solution to improve the system interoperability is by adopting SOA technology. Indeed, the success of interactions among the systems depends on how well the service interfaces are exposed. The stability of the service interfaces in solving the SI problem has been the subject of many researchers such as in (L'Amrani et al., 2020; Paniagua et al., 2019; Madni & Sievers, 2014; Masethe et al., 2016; Teo and Kadir, 2006; Henkel & Zdrakovic, 2005). To address the interoperability issues (Rezaei et al., 2014), interfacing processes need to be made more easily by defining components, subsystems, processes, and interfaces that affect the SI architecture at the initial design stage. The importance of service interfaces in integration process utilises this study to focus on the interoperability of the legacy systems involved.

In this paper, three prominent approaches related to Service Interface Design (SID) are proposed and discussed, which are method-oriented, message-oriented and resource-oriented. Each approach with its own interface model is illustrated to show the interface solution for high interoperability in heterogeneous SI.

BASIC CONCEPT OF TRADITIONAL SYSTEM INTEGRATION PATTERNS

In this section a few patterns of traditional SI are briefly reviewed. The purpose is to look at the methods and the mechanisms of this traditional SIP used for sharing data or information.

File-Based

File Transfer Protocol is the most common method of file transfers in the case of two applications running on two different machines. The text file is the most common type for file-based data sharing because the character is represented by one byte in almost all operating systems and languages (Kazman et al., 2013). This method allows different applications running on the same machine to read and write to the same file. Figure 1 shows how file sharing over a network.

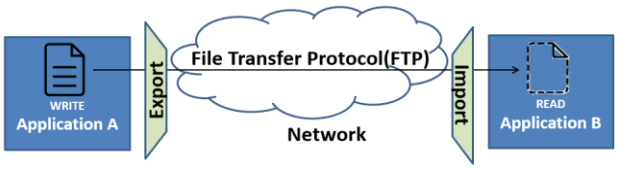

Figure 1: File-based data sharing method

Common Database

Common database method is quite similar with file-based method in term of data sharing where two applications running on two different machines able to read and write from/to a common database (Kazman et al., 2013). The database in most cases always runs on its own machine. Figure 2 shows how the common database method sharing data.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

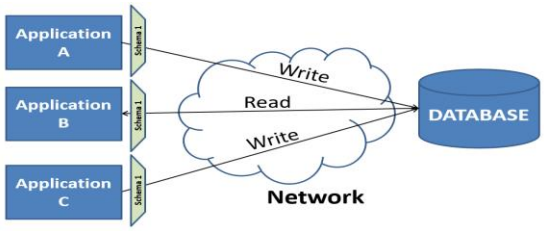

Figure 2: Common Database sharing method

Integration is achieved by storing all data in a shared database where it defines the schema of the database to handle all the needs of the different applications. The widespread use of SQL based relational databases as the common database become very popular for application integration. In addition, SQL is supported by almost all development platforms.

Remote Procedure Call (RPC)

RPC is a function-oriented interface. It is also known as client/server and 2 tier architecture. It introduced important concepts and features of functions sharing (Hohpe & Woolf, 2011). RPC involves synchronous function calls only where the calling code is allowed to do further work until the function returns. It has three types of function calls involved stated as below:

1. Local calls.
2. Restricted remote calls involving two applications (on the same machine).
3. RPCs between two applications (on different machines) connected by a network.

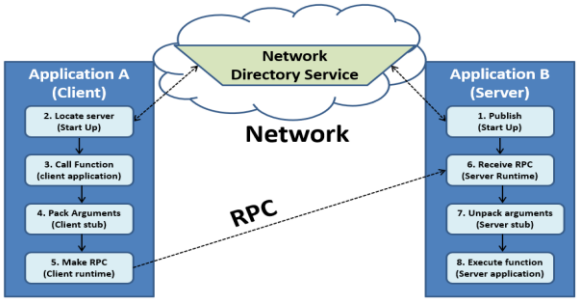

Figure 3: Remote procedure calls between different hosts

In this method, the application calling the function is called client and the application where the function resides is called server (as illustrated in Figure 3). Thus, this method involves service concept and it may be considered as the beginning of the services-based integration pattern.

Distributed Objects – Object Request Broker (ORB)

Distributed objects (DO) are a method with language platform independence which means applications can share functionality even though it is written in different programming languages and different platforms (Hohpe & Woolf, 2011). The distributed objects extend the concepts of classes and objects introduced by object-oriented programming (OOP). Java and C++ languages are samples of OOP. Classes are user-defined constructs that encapsulate functionality and data related to a certain entity. OOP includes inheritance which leads to code reuse, and polymorphism that allows functions performing similar work can have the same name.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

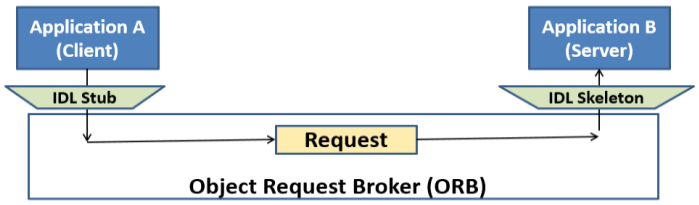

Figure 4: Object Request Broker Architecture (CORBA)

Three models of distributed objects are Microsoft's Distributed Component Object Model (DCOM), Java Remote Method Invocation (RMI), and Common Object Request Broker Architecture (CORBA).
Like RPC, DO also employ synchronous interaction. Client application can do further work until the server application completes its work and returns control to the client application (as illustrated in Figure 4).

Messaging

Three basic elements of messaging system are message, queue, and endpoint. Applications are communicating through queues. There are two type of basic queues which are point-to-point, and publish-and-subscribe. Point-to-point allows message to be given to one receiver only (as illustrated in Figure 5). In publish-and-subscribe, any number of receivers can get and act on a message (as illustrated in Figure 6).

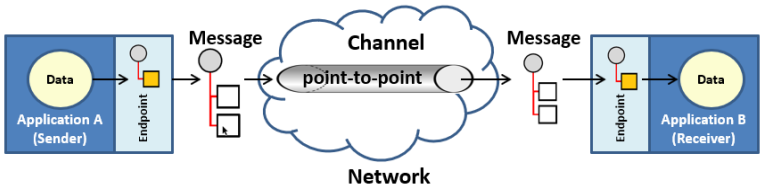

Figure 5: Point-to-point Queue

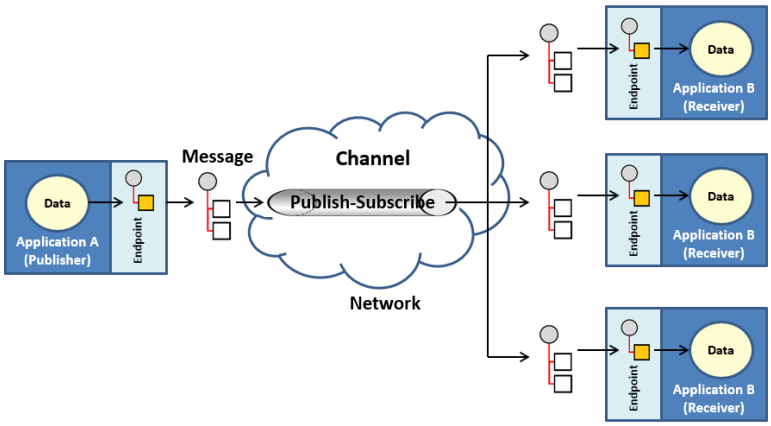

Figure 6: Publish-and-subscribe Queue

Messaging system is also called Message Oriented Middleware (MOM). One of the important features of MOM is guaranteed delivery. For example, a message sent to a target application is unable to be delivered due to network is down, the messaging system will try and try again until the message is delivered (which is the network is available).

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

Messaging is an asynchronous type interface. In this method, a service consumer who sends a message to a service provider is not waiting for the response. It allows the service consumer (client) application to perform other work while the service provider (server) is completing the request from the client.

SERVICE INTERFACE DESIGN APPROACHES

This section defines the SID approaches covering method-oriented service, message-oriented service and resource-oriented service. An overview of SID approach architecture is illustrated to show the proposed components involved in interfacing with legacy application using traditional SIP.

Service interface refers to a set of defined operations that enable the communication or integration process occurs. The "service interface" term is commonly used in the service-oriented environment that refers to communication point between components. It is one of the key elements for interaction in systems integration. It defines a way in which other elements can interact and exchange information with the service. Many approaches to SI with regards to SID have been identified by previous researchers (Papazoglou & Heuvel, 2007; Henkel & Zdrakovic, 2005; Teo and Kadir, 2006).

Service Oriented Architecture (SOA) with Web Services (WS) concepts has been chosen as a system architectural for SID approaches. Several research regarding SOA have decided to address the implementation of WS as a technology used to implement the principle of service orientation (Awang, 2015; Wei et al., 2011; Shej & Cico, 2011; Zhao et al., 2011). A well-designed service interfaces concept used in WS is said to be an important element to make ease of SI (Jie et al., 2010).

Most definitions of SOA identify the use of WS using Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI) in its implementation. SOAP protocol is used for accessing a WS (UDDI, 2021). WS can be published and discovered using UDDI. WS interactions share a common WSDL file. Figure 7 shows typical relationship of WS.

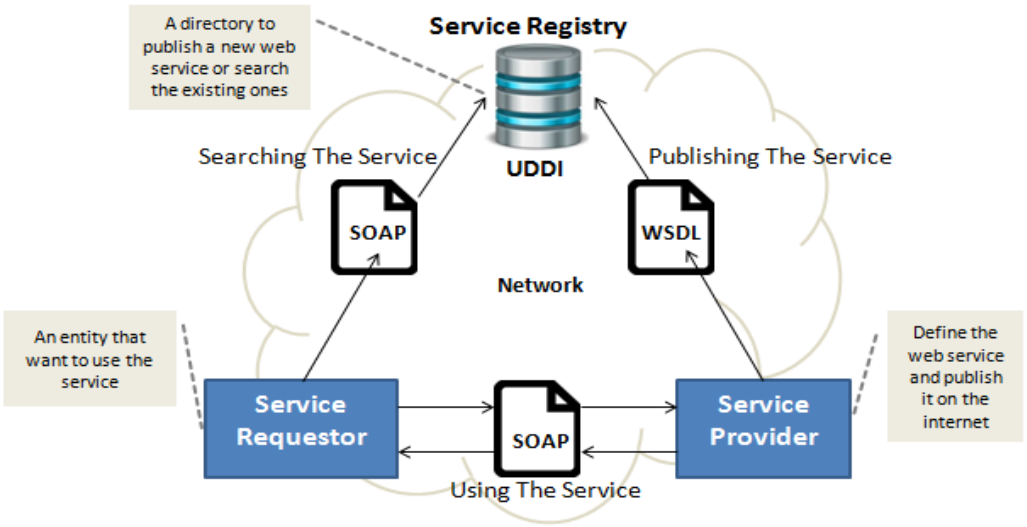

Figure 7: Web Service Typical Relationship Lifecycle

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/4.0//)

The following sections describes three approaches to SID proposed to provide solution for upgrading interoperability of traditional SIP. The three SID approaches are method-oriented interface, message-oriented interface and resource-oriented interface.

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 (Teo and Kadir, 2006). Figure 4 shows one of the technologies of method oriented.

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) (Krzyzanowski, 2010).

In web service application, a specific operation on a service point with input arguments is invoked by service consumers. Description of the operations which the interface supports including their parameters is contained within WSDL files. But the use of XML schema to describe the parameter structures is optional. In method-oriented design, WSDL plays a major role whereas XML a minor.

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 (Henkel and Zdrakovic, 2005; Teo and Kadir, 2006).

Figure 5 and Figure 6 show the technologies of message oriented. 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 WSDL plays a minor role.

Resource-Oriented Interface

The resource-oriented interface or constrained interface 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 (Dar et al., 2015; Vettor, 2016) and content-oriented interfaces are used in lieu of constrained interfaces.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

SID Architecture

Figure 8 illustrates the proposed architecture of SID approaches for legacy application involving traditional SIP. Based on this architecture, any traditional SIP used by the legacy application has to be converted into relevant SID to upgrade its interoperability with any web-based client applications.

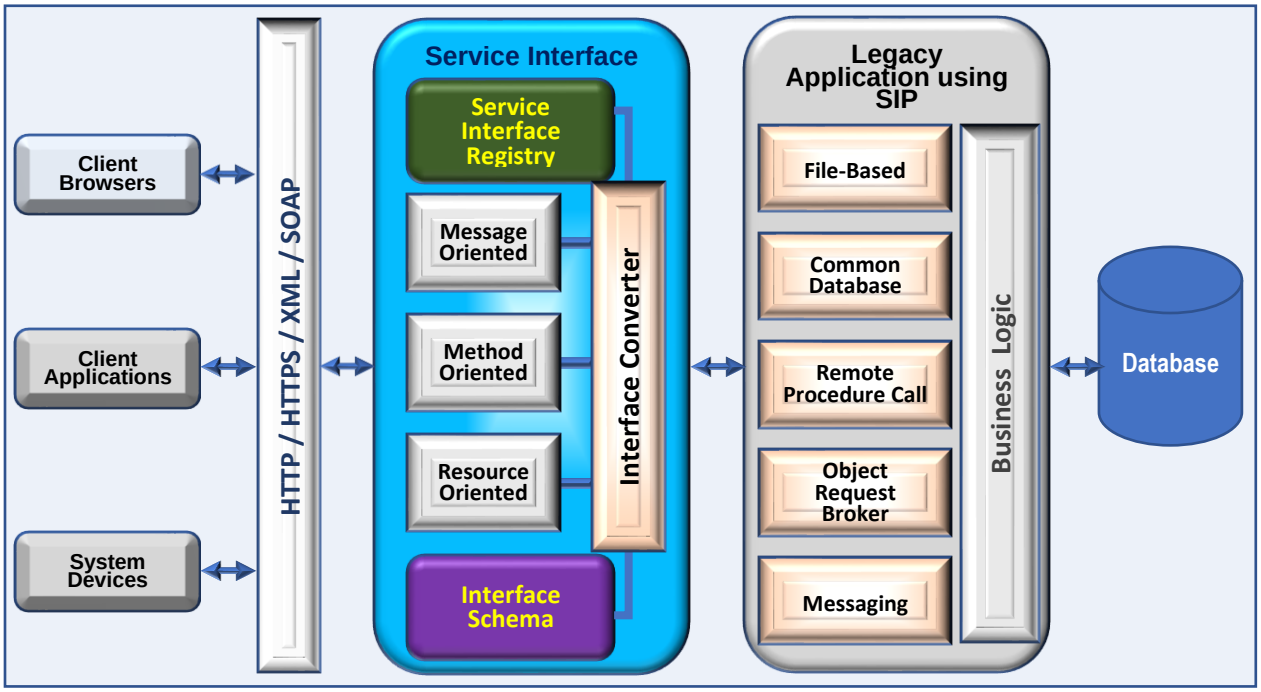

Figure 8: SID Architecture

The first step is to define service interface. The legacy application has to define and establish type of data it requires to publish and share to other systems or applications. Service interface will define interface name and schema, and register them in the Service Interface Registry. The interface schema stored in the Service Interface Registry will also be published in the UDDI.

The type of service produced depends on the type of SIP and according to the compatibility between SID and SIP. For example, interfaces that use RPC are more suitable to be converted to Method Oriented Service.

Summary of SID Approaches

Table 1 provides example of definition of interfacing between the three prominent SID approaches and traditional SIP. SID approaches are suggested to be clustered in a single interface model to provide high interoperability in heterogeneous SI.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

Table 1: Interoperability of interfacing between SID Approaches and traditional SIP

Traditional SIP	Service Interface Design Approaches		
	Method-Oriented	**Message-Oriented**	**Resource-Oriented**
File-Based	Get_File (source, filename)	Not Applicable	(https://lpt.com.my/report.html)
Common Database	Not Applicable	Not Applicable	GET /restsql/res/staff?position="operator"
Remote Procedure Call	cctv_cam (directory, location_id, status)	Not Applicable	(https://lpt.com.my/cctvreport.html)
Object Request Broker	camera (hostname, directory, location_id, status)	Not Applicable	(https://lpt.com.my/fatalreport.html)
Messaging	ViewReport (text message)	ViewReport (text message)	(https://lpt.com.my/report.html)

As summarized in Table 2, each approach has advantages and disadvantages, but it does not mean that architect has to choose the ideal approach as the solution to their SI. As long as the approach(es) can meet the demand of the integration requirements, either with single or combination of approaches, it can be considered has met the objective or purpose of integration. Selection of a single approach to SID of SI is not practical due to various systems with various technologies involved in the SI.

Table 2: Summary Comparison of SID Approaches

Method-Oriented Interfaces	Message-Oriented Interfaces	Resource-Oriented Interfaces
▪ RPC type. ▪ Service consumers have to know the service interface definition exactly. ▪ Changes on the interface will need the service to be updated (tightly coupled). ▪ Service consumers invoke an application-specific operation on a service endpoint. ▪ Description of operations that the interface supports, and their parameters are contained within WSDL files. ▪ XML plays a minor role whereas WSDL a major.	▪ Interface is fixed. ▪ Message structure is changeable. ▪ All the messages are described by using XML schema. ▪ Difficult to interpret and understand the functionality provided by a service. The structure of the messages that the service can handle need to be examined in order to understand the functionality of the service. ▪ XML plays a major role whereas WSDL a minor.	▪ Interface adheres to a fixed set of standardized operations. ▪ HTTP is one of the resource-oriented interfaces. ▪ Possible to build large distributed systems. ▪ Interface is standardized does not need to be updated.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/4.0//)

CONCLUSION AND RECOMMENDATION

In traditional SI, application developers are more familiar with single type of interface that able to integrate with limited integration pattern or systems. In such case, the integration design is scoped within the context of single SID approach. For example, method-oriented services can only talk with few SIPs such as file-based and remote procedure call.

Selection of a single approach for heterogeneous SI is not practical due to various systems with various technologies involved in the SI. Hence, the SI involving SIP demands a service interface mapping mechanism which can mediate the interaction between SID and SIP during the integration process.

For future works, a mediator approaches could be a focus on providing access to specific functions or services for heterogeneous systems.

REFERENCES

Awang, N. (2015). MIPAF: A Policy-based Middleware Framework to Control Nedative Effects of Software Evolution. Journal of Theoretical and Applied Information Technology. Vol.70 No.3.

Buyle, R. (2017). Towards interoperability in the public sector. Presented at the ISWC2017, the 16e International Semantic Web Conference. Vol. 1931, pp. 1-8.

Dar, K., Taherkordi, A., Baraki, H., Eliassen, F., & Geihs, K. (2015). A resource oriented integration architecture for the Internet of Things: A business process perspective. Pervasive and Mobile Computing, 20, 145–159. https://doi.org/10.1016/j.pmcj.2014.11.005

Henkel, M. & 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

Hohpe, G. & B. Woolf (2011). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. The Addison Wesley Signature Siries.

Jakimoski, K. (2016). Challenges of Interoperability and Integration in Education Information Systems. International Journal of Database Theory and Application, 9(1), 33–46. https://doi.org/10.14257/ijdta.2016.9.2.05

Kazman, R. (2013). Understanding Patterns for System-of-Systems Integration. Software Engineering Institute.

L'Amrani, H. L. '. A., El Bouzekri El Idrissi, Y., & Ajhoun, R. (2020). Intermediary Technical Interoperability Component TIC Connecting Heterogeneous Federation Systems. Machine Intelligence and Big Data Analytics for Cybersecurity Applications, 919, 521–539. https://doi.org/10.1007/978-3-030-57024-8_24

Madni, A. M., & Sievers, M. (2014). System of Systems Integration: Key Considerations and Challenges. Systems Engineering, 17(3), 1–10. https://doi.org/10.1002/sys.21272

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA
(https://creativecommons.org/licenses/by-nc-sa/4.0//)

Masethe, H.D., Adewumi, A.O., & Masethe, M.A. (2016). The Scoping Review of Integration of Heterogeneous Healthcare Systems. Proceedings of the International Multi Conference of Engineers and Computer Scientists 2016, Vol I.

Paniagua, C., Eliasson, J., & Delsing, J. (2019). Interoperability Mismatch Challenges in Heterogeneous SOA-based Systems. IEEE International Conference on Industrial Technology (ICIT), Melbourne, Australia, 2019, 788–793. https://doi.org/10.1109/ICIT.2019.8754991

Papazoglou, M., & Heuvel, W.-J. H. (2007). Service oriented architectures: Approaches, technologies and research issues. The VLDB Journal, 16, 389–415. https://doi.org/10.1007/s00778-007-0044-3

Rezaei, R., Chiew, T. K., Lee, S. P., & Shams Aliee, Z. (2014). Interoperability evaluation models: A systematic review. Computers in Industry, 65(1), 1-23.

Roshen, W. (2009). SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-based Application Integration. McGraw-Hill, Inc.

Shej, A., & Cico, B. (2011). Integration of Applications Based on SOA in Government Institutions. 1st International Symposium on Computing in Informatics and Mathematics (ISCIM 2011).

Teo, H. M. & 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).

UDDI (2021). Universal Description, Discovery, and Integration (UDDI). Version 3.0.2 Specification. Retrieved Jan 15, 2021, from https://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/rwsu_terminology.html

Vettor, P. (2016, December). A Resource-Oriented Architecture for Integration and Exploitation of Linked Data. Hardware Architecture [cs.AR], Universityé de Lyon. https://tel.archives-ouvertes.fr/tel-01422057

Wei, N., Song, M. N., Xu, K., & Jiang, C. (2011). A Novel Web Service Architecture Based on REST. Advanced Materials Research, International Conference on Smart Materials and Intelligent Systems, SMIS 2010, 143-144.

Zhao, X., Liu, E., &Clapworthy, G. J. (2011). A Two-Stage RESTful Web Service Composition Method Based on Linear Logic. Paper presented at the 2011 IEEE Ninth European Conference on Web Services.

Copyright© 2021 UiTM Press. This is an open access article under the CC BY-NC-SA (https://creativecommons.org/licenses/by-nc-sa/4.0//)