Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho

Vol. 6, No. 1, pp. 35–41, Mar.-2018

p-ISSN: 2410-7549
e-ISSN: 2414-6943

WEB APPLICATIONS AND WEB SERVICES: A COMPARATIVE STUDY

Qusay I. Sarhan [a,*] and Idrees S. Gawdan [b]

[a] College of Science, University of Duhok, Kurdistan Region, Iraq – (qusay.sarhan@uod.ac)
[b] Curriculum Development Center, Technical College of Engineering, Duhok Polytechnic University, Kurdistan Region, Iraq.

Received: Jul. 2017 / Accepted: Mar., 2018 / Published: Mar., 2018 *https://doi.org/10.25271/2018.6.1.375*

ABSTRACT:
Software components that rely on the Internet in order to be accessed and used cover many aspects of our daily activities including email checking, weather checking, purchase ordering, and Facebook logging. Nowadays, these components are considered one of the most valuable and fundamental rights in human's life. However, accessing and using the functionality of such components are performed using two techniques: web applications or traditional web services. Determining which one of the techniques is better suited for delivering the functionality of an Internet-based software component is not an easy task to decide. Therefore, this paper aimed to make a comparative analysis and study of both techniques alongside many directions to help web developers make the right choice to deliver the functionality of Internet-based software components. This is achieved by providing them with a set of requirements that have been proposed by the authors. The proposed requirements clarify a number of misunderstandings and issues peculiar to both techniques. Furthermore, this paper has provided a comparative table of both techniques. To the best of our knowledge, so far there is no comprehensive comparative study has been conducted in this context which was the rationale for the authors to carry out this study.

KEYWORDS: Internet-Based Software Components; Web Applications; Web Services; Internet of Things; Comparative Study.

1. INTRODUCTION

Internet-based daily life is obvious for everyone and it is one of the most prominent characteristics of the current era in which people live. A variety of Internet-based software components are available to make our life easier and smarter. The exponential increment of people and devices located at different places and connected to the Internet using different means of connectivity made them an important part of the Internet (Gubbi *et al.*, 2013). The evolution in Internet technologies to meet the demands of this increment led to the emergence of new application domains such as Internet of Things (IoT). The IoT connects everyday objects around us to the Internet and manages this kind of connectivity (Khan *et al.*, 2012). Besides, it provides a wide range of Internet-based software components that help us to achieve many things in an easy and smart way. Knowing a traffic state near your work place using your mobile phone is just a very simple example about using the IoT. However, this new interaction raises a question of: How can users access and use the functionality of different Internet-based software components exist due to such recent domains. In other words, which technique has to be selected and used by web developers in order to deliver the functionality of these components to beneficiaries? Both web applications and web services with their notable differences represent the techniques of accessing and using Internet-based software components. However, users see both techniques completely the same in terms of work nature, technical features, and consider no difference exists between them at all. Usually, they need to accomplish some Internet-based tasks, such as checking the daily weather or checking bank account either using their personal computer or any other computing device that has the ability to connect to the Internet. They care only about getting the tasks they need done regardless of what is going on behind technical scenes. Therefore, the technical buzzer-words are not important for them. This point of view is not applicable for web developers; they focus and care more about the technical aspects. They need to know everything about these techniques including how to select the best technique, how to use it properly, and many other important aspects. In literature, several authors presented many comparison studies of web applications and web services against standalone applications or distributed applications and the ability of converting web applications into web services using wrapping and reverse engineering techniques (Yu *et al.*, 2007, Cook & Barfield, 2006, Torchiano *et al.*, 2009, & Lorenzo *et al.*, 2007). However, there is no study that has compared both techniques and provided a set of requirements that help developers select the most suitable technique for a specific usage. In this paper, we aim to contribute to the following:

- Provide a comparative study of both web applications and web services alongside many directions. On top of that, a set of requirements have been proposed by the authors to help developers go for the right choice. The proposed requirements are organized in many categories to make them easy to be remembered and followed.
- As this comprehensive study provides deep understanding of many aspects regarding both techniques. We also hope to provide an integrated comparison guideline as an essential point to guide whoever wants to work in this context.

The remainder of this paper is organized as follows. Section 2 presents the problem of this study and its space. On top of that, it describes the requirements that have been proposed by the authors to address the problem. In Section 3 and 4 respectively, both web applications and web services have been compared

* Corresponding author

according to the proposed requirements. Finally, we provide a table of comparison of both techniques as the summary of our contribution and we conclude this study in Section 5.

2. THE PROBLEM AND ITS SPACETE

As this paper considers the problem of choosing the appropriate technique for delivering the functionality of Internet-based software components to beneficiaries, the authors proposed a set of comparison requirements to help in this respect. These requirements vary across a number of different aspects as described below. In the subsequent sections, we will refer to these requirements in detail to address our problem.

- **Terminology and Usage:** this refers to the technical definition of the technique with its general usage. This requirement helps to clarify the misunderstanding of each technique in terms of definition.
- **User Interfaces:** these specify whether the technique provides a user interface to its users in order to allow them to interact smoothly with its functionality or not. In case the technique supports a user interface, it is important to specify which type of interface is supported by the technique.
- **Structure View:** this specifies the external elements of the technique, the relationships among these elements, and the ways they communicate with each other. Generally, it does not provide information about the internal functionality of the technique.
- **Interaction Models:** these specify how the software functional units of the technique interact with each other in order to exchange data among themselves. Besides, they help to determine if the used technique supports the collaborative work or not.
- **Building Blocks:** these represent the internal components of the technique. For example, the internal code segments that collaborate with each other to deliver the software functionality. It is worth mention that this requirement shows how complex is the used technique.
- **Testing:** this specifies how the technique can be tested in order to find different types of errors. Also, it determines the well-known testing strategies related to each technique.
- **Accessibility and Target Platform:** these specify if the technique provides cross-platform compatibility or not. On top of that, it determines the barriers against interoperability.
- **Working Nature:** this specifies if the technique provides synchronous/asynchronous request-response operations or not. This requirement can be used to determine the flexibility of a specific technique in terms of supporting different types of request and response operations.
- **Infrastructure Control:** this specifies if the technique is integrated into the user system environment or not. On top of that, it determines if there will be any bad impact on the user's system in case of any failure.
- **Release Control and Updating:** this specifies if the user will be involved in case of any updating process or not. Besides, it determines the simplicity of the updating process using a specific technique.
- **Composition and Reusability:** this determines if the technique allows combining some functionality into each other to provide more powerful functionality or not. Furthermore, it determines if composition and reusability can be achieved in runtime or not.
- **Complexity:** This requirement helps to provide the complexity level of developing or using a specific technique. Furthermore, it provides the factors that increase or decrease the complexity level alongside many directions.

3. WEB APPLICATIONS

3.1 Terminology and Usage

A web application is any piece of code that can be accessed using web browsers, such as Mozilla Firefox and Google Chrome running on the client's machine. It depends on web browsers to be executed and then to deliver its functionality to the client. In this case, the web browser acts as the universal client for any web application. Generally, web applications are developed using browser-oriented programming, scripting, and styling languages/frameworks alongside server ones such as Hybrid Text Markup Language (HTML), Cascading Style Sheet (CSS), JavaScript, and Personal Home Page (PHP). However, web applications are very popular due to their notable features including the ease of use (humans are the users) and the ability to update their contents without installing any software on potentially thousands of clients' machines (Lam, 2011).

3.2 User Interfaces

A web application provides users the ability to interact smoothly with its functionality through different types of user interfaces. User interfaces represent the only visible parts for users and the most important parts of any web application. That is because they determine how easily users can use a web application. Generally, user interfaces are divided into five types as follows:

- **Command Line Interface (CLI):** It is the simplest one among other types of user interfaces. In the CLI, the user interacts with the web application by typing commands in a specific screen using the keyboard and the application provides back the output by printing it mostly on the same screen. However, it is worth mentioning that web applications nowadays scarcely utilize the CLI to deliver their functionalities to users. Comparing the CLI to other types of user interfaces, the CLI is not considered as a user-friendly mechanism for application-user interaction.
- **Graphical User Interface (GUI):** The GUI presents a user-friendly mechanism for interacting with any application. This interaction is performed via a set of graphical components including menus, buttons, labels, and many others. Besides, it provides a distinctive look and feel to attract users. Nowadays, web applications exceedingly utilize the GUI to deliver their functionalities to users.
- **Zoomable User Interface (ZUI):** The ZUI is a special type of the GUI. With the ZUI, users are able to see more detail or less by changing the scale of the view area. In other words, the user can browse almost everything simply by zooming in and out. Web applications that deal with data visualization and map processing utilize the ZUI to deliver their functionalities to users.
- **Voice User Interface (VUI):** With the VUI, users interact with the application through voice/speech based commands in order to utilize its functionality. The most important feature of the VUI is providing hands-free and eyes-free application-user interaction. Web applications that are designed to help people with some disabilities utilize the VUI to deliver their functionalities to them.
- **Activity User Interface (AUI):** It is also called gesture user interface. With the AUI, different types of gestures can be originated from a human face or hands to then be processed and recognized as commands to a web application. Web applications that deal for example with face recognition utilize the AUI to deliver their functionalities to users. In such applications, using some hardware devices, such as cameras and sensors in are commonplace.

The aforementioned types of interfaces help users of web applications to send/receive data to/from a web application over the Internet using their preferred web browsers. The data then processed and presented to them within their browsers as information (Tesarik *et al.*, 2008). However, selecting the appropriate user interface depends mainly on the aim and domain of a web application.

3.3 Structure View

A web application consists of one or more web pages that are created usually by using a various number of web programming and scripting languages. These pages contain a combination of static and/or dynamic contents including text, images, and code that can be run on servers or web browsers. Users can access these pages using their web browsers. Web applications can reside on servers that have the ability to handle user requests and to provide back the required responses. Besides, they can use multiple servers in the network in order to deliver their functionalities. Generally, users are not aware that the required task requested by them might be distributed across multiple servers (Marinho *et al.*, 2011). Figure 1 shows the conceptual model of a web application.

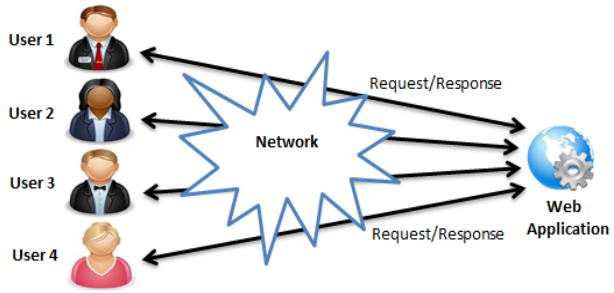

Figure 1. Conceptual model of a web application

3.4 Interaction Models

Interaction among different web applications in runtime does not exist. For instance, two web applications cannot interact or collaborate with each other in order to perform some collaborative tasks. This is due to the fact that web applications are human-oriented applications. In other words, they are meant to be used by humans only. As a result, each web application is independent from others and data cannot be exchanged dynamically in runtime among them (Moreno & Vallecillo, 2005). It is worth mentioning that missing the interaction among web applications means developing a web application always starts from scratch. In other words, web developers are not able to re-use some existing software components that are already developed by others. This leads to develop the same components with the same functionalities once again even if they exist somewhere on the network. Thus, the main drawbacks of this situation are cost, affordance, and time consumption.

3.5 Building Blocks

The main building blocks of any web application are typically composed of three blocks: Presentation, Business, and Data Access (Mašovi, 2012) as shown in Figure 2.

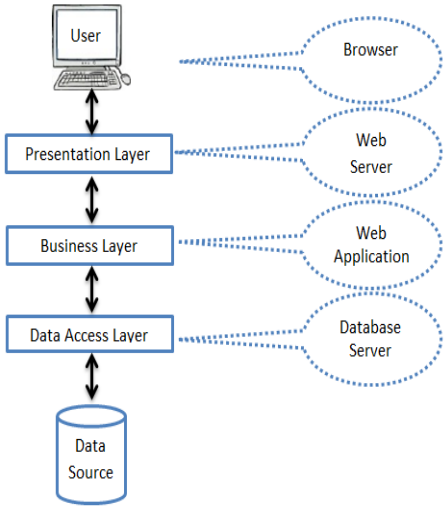

Figure 2. Typical building blocks of a web application

- **Presentation Layer:** Is the topmost level of the application and represents the interface to its users. Besides, it can be reached through web browsers and is used to translate user requests and responses to something the user can understand easily.
- **Business Layer:** Is used to coordinate the application processes, logical decisions, evaluations, and calculations. Moreover, it processes and passes data and information between the two surrounding layers.
- **Data Access Layer:** In this layer data and information are stored and retrieved from a database. Information is then passed back to the business layer for more processing, and eventually returned back to the user.

3.6 Testing

Internet-based software components that are deployed as web applications or web services and the struggle to make them work properly cause a variety of errors that reduce the required level of functionality. These errors can be in many forms including miss-implemented functionality, performance errors, security errors, and errors that cause the entire web application or web service to fail (Repasi, 2009). The general testing strategies of both techniques include functionality, usability, compatibility, and performance testing approaches. Despite that, some testing approaches are unique and peculiar to each technique. However, testing of web applications includes the following main approaches (Lucca & Fasolino, 2006, Doğan, 2014):

- **Web Page Death:** As web applications are combined with web pages to explore their interfaces to users, these pages have to be tested intensively. Testing of pages is very important to check the status of a specific page if it is a dead or active. The dead page is any page that does not work properly, cannot be reached, or searched within the website that contains a web application. On the other hand, the active page is any page that works as required to present the functionality of its embedded web application to be used by users.
- **Web Page linkrot:** A web link that does not work is called broken link or linkrot. However, linkrot occurs when the target of the reference is misused or no longer exists. As a result, users lose the ability to access a specific web application in order to use it.
- **Web Page Redirection:** It means the link of a web page that contains a web application is working properly but not pointing to the required reference target; could be pointing to

another unrequired target. Therefore, redirection of web pages should be checked repeatedly to find any invalid redirection.

- **Field Validation:** Input validation for each field in any web page that contains a web application should be performed rigidly. Such input values of these fields may be fed to a web application in order to deliver its functionality in a proper way. In many cases, web applications require different types of user data. For example, a user name has to be entered in a specific field and in a specific format in order to reach the next step of an application's functional work. Therefore, missing the required inputs or providing them with no validation may lead to a fail in the whole web application. However, negative testing can be used in such similar situations to perform field validation. For example, wrong inputs such as using a person's name combined with numbers or with age field some letters can be entered rather than numbers to apply field validation.

3.7 Accessibility and Target Platform

Web applications provide fully interoperability or cross-platform compatibility feature. Therefore, they can be accessed and used by: any software platform (including Windows, Linux, and Mac), any computing device, and at any location. Interoperability makes web applications accessed by the broadest audience using heterogeneous platforms for the least effort. The concept of cross-platform makes the development process of web applications easier rather than developing applications tied up to a specific platform. Developing applications to a specific platform limits its using in an obvious way (Fernandes, 2012). Besides, any big change occurs in that platform means the whole application has to be changed accordingly which is not feasible and time consumed. Users simply can reach any web application by using its URL through their web browsers which instantly deliver the application's functionality to their devices.

3.8 Working Nature

Web applications are known for their synchronous nature. In other words, users issue their requests to perform some tasks and expect the instant responses to be sent back. For example, when a user request a Gmail account login after providing the required information (usually are username and password), he/she expects to be logged in directly. However, synchronization requires Internet connectivity to get data exchanges from the server side and update the data in the user side. Mostly, this is achieved without storing a web application's data on the local host of the user.

3.9 Infrastructure Control

Web applications are not integrated into the infrastructure of the user's system. They reside in foreign infrastructures that could be located anywhere in the world. All user requests are passed to the server side in order to be processed without requesting to make any change in the user side. Therefore, any fault or crash in the application or server does not affect in any way the user's system.

3.10 Release Control and Updating

Developmental releases of web applications do not affect users in any way. Everything will be performed without involving them. Generally, updating a web application involves fixing a defect or adding a new feature. Therefore, when it comes to update a web application that is already deployed; users do not need to follow any updating procedure or make some modifications in their environment in order to get the latest release of that web application (Danesh *et al.*, 2011). As a result, every time they request a web application via their browsers they directly get the latest and updated release of it.

3.11 Composition and Reusability

In web applications, users cannot combine some ready-made applications into a single application to make it more valuable and powerful (Moreno & Vallecillo, 2005). This is because of that web applications cannot interact dynamically in runtime with each other as mentioned in a previous section. However, as web applications can be accessed through web browsers, the combination of two web applications for example can be achieved by opening them in two separated tabs or in two different web browsers. The first tab can be used to enter some data to be processed by one of the applications. Then, the produced results can be copied and fed into the second tab for further processing. However, users need to repeat this process many times if they want to apply the same processing for a long period of time. As noted, composition process in web applications is completely human-based and consumes a lot of time and effort. It is worth mentioning that as web applications do not support composition, they do not support software integration in runtime. However, integration can be achieved by opening all the required applications then manually exchange data among them which is not an efficient way of making benefit of using different applications. Regarding reusability, using web applications does not help in this respect, for developing a new web application the required programming tasks must be developed from scratch even if they are already exist somewhere else on the Internet.

3.12 Complexity

Developing web applications does not require a lot of complexity comparing to web services. The development process does not involve using external entities (excluding application servers). Besides, it does not rely on heavyweight protocols for communicating and exchanging data as everything is handled by web browsers. On top of that, there is no integration processes between the user side and the application side. All the aforementioned factors decrease the complexity level of developing such applications.

4. WEB SERVICES

4.1 Terminology and Usage

A web service is a number of independent functional components that allows different machines to interact and collaborate with each other through a network to achieve a common goal (Yu *et al.*, 2007). To use a web service properly, clients require some information about the service including its name, its location, and many others. Usually, web service engines and repositories are available to help developers searching and then locating web services in order to use them. For example, clients can search for a specific web service using keywords and then from the results list any service can be invoked.

4.2 User Interfaces

Web services do not provide users with any interactive interfaces as in web applications in order to enable users to interact with their functionalities. This is due to that web services meant to be used only by machines not humans. As there is no user interfaces supported, they share business logic, data, and processes through programmatic interfaces across a network called application interfaces instead of user interfaces

(Cook & Barfield, 2006). However, developers can use application interfaces to embed a specific web service to a user interface, a web page, or an application in order to extend its functionality to users.

4.3 Structure View

Ready-made web services available on the network allow distributed software components that are language and platform independent to be accessed and used by different applications across the Internet. Applications can perform some of their tasks by making use of these ready-made services (Yu *et al.*, 2007). Figure 3 shows the conceptual model of web services.

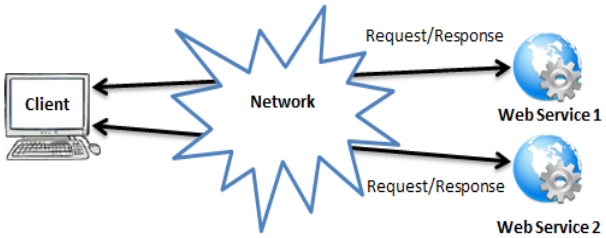

Figure 3. Conceptual model of web services

Web services use industry-standards protocols and technologies including Hyper Text Transfer Protocol (HTTP) to provide standardized way of communication among different entities. A web service is usually provided by a server (service provider) and can be accessed and used by a client (service consumer). The server and client exchange request (method name, parameter list) and response (return value) messages in order to interact with each other smoothly.

4.4 Interaction Models

Web services interact with each other by exchanging Extensible Markup Language (XML) based text messages which are human-readable. The XML message is a well-constructed hierarchy of XML tags that together describe a set of data fields sent or received by different web services. Each XML message consists of two sections: header and payload. The header section stores control information about the message, whereas the payload section contains the actual content of the message. However, web services use two types of interaction models in order to exchange XML message between each other (Maximilien & Singh, 2005) as follows:

- **Peer to Peer (P2P) Interaction Model:** In this model, two web services exchange request/response messages using HTTP over the Internet. Often, request messages could be for a Remote Procedure Call (RPC) to invoke the functionality of a piece of code or an XML data document to get some kind of data. Whereas, response messages could be a computation result from the RPC or XML data document. However, request/response messages are managed by web services using Simple Object Access Protocol (SOAP). Figure 4 shows the P2P interaction model between two web services.

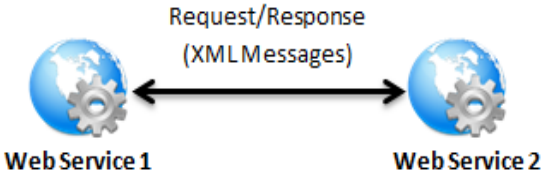

Figure 4. P2P interaction model

- **Multilateral Interaction Model:** In this model, more than two web services exchange messages among themselves. Multilateral interaction is achieved by aggregating multiple P2P interactions as shown in Figure 5.

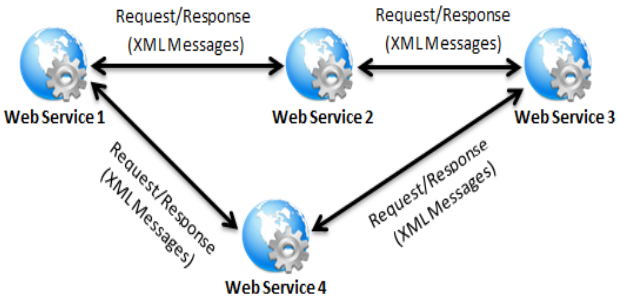

Figure 5. Multilateral interaction model

4.5 Building Blocks

To utilize web services, a number of blocks are required to work and collaborate with each other. The SOAP, Web Services Description Language (WSDL), and Universal Description/Discovery/Integration (UDDI) are the main building blocks of any web service (Lee, 2014, Anass & Ahmed, 2017) as shown in Figure 6.

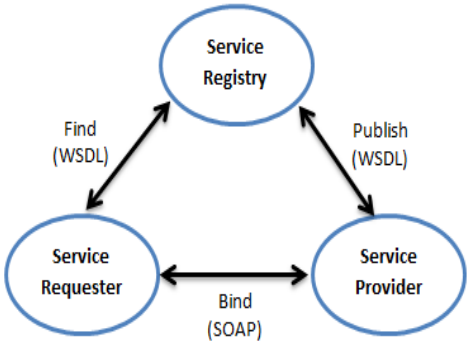

Figure 6. Typical building blocks of a web service

- **The SOAP:** Is a simple XML based protocol that defines the structure and data type of messages used for communication in web services. A SOAP message consists of an envelope with a message body and an optional header. The SOAP body contains the actual message content and the SOAP header is used for passing control information. However, SOAP messages typically travel over HTTP which is the standard network protocol.
- **The WSDL:** Is a document that describes web services alongside many directions. For example, it describes the format of requests that web services handle, parameters to be supplied to each request, and the format of responses. The WSDL document of a web service is stored in a text file with the extension of .wsdl. Besides, it can be located typically on the same server where the web service itself is deployed. Web service developers use WSDL documents to describe the functions that web services provide and how other programs can access and use those functions. Usually, web service requesters analyze WSDL documents to know the functionalities of the provided web services, their locations, and how to invoke them properly.
- **The UDDI:** Is a mechanism for registries intended for storing and publishing descriptions of web services in forms of WSDL documents. Web service providers use UDDI to register their services with all the required information regarding how they can be located and invoked by service requesters.

4.6 Testing

Testing of web services includes two main approaches (Sharma *et al.*, 2012) as follows:

- **The WSDL Document Validation:** While creating the WSDL document for a web service, interfaces alongside other important information have to be correctly and precisely described. Besides, service requesters must conform to contracts specified in WSDL documents in terms of message content, binding to the transport layer, and many others. It is worth mentioning that mistakes and ambiguous descriptions in WSDL documents can affect the whole communication process among services and may lead to communication failure.
- **Publish, Find, and Bind (PFB) Testing:** In order to use a web service, the service has to be already published properly. Therefore, service providers register their services to service brokers/servers. Then, these brokers advertise about the registered services via different types of information regarding each service in brokers that can be found through search methods. The search process helps service requesters to find the needed services and then bind to them for consuming their functionalities. In this respect, the PFB testing must answer the following questions:
  - Are services able to register themselves to brokers?
  - Can applications or other web services find and bind to registered services?

4.7 Accessibility and Target Platform

Web service interoperability is a big issue especially when a service requester encounters problem while invoking a method provided by a service provider. Or, when it does not really understand messages sent by a service provider. These issues usually happen by misusing different types of prerequisites that are exposed by a service provider or requester environment. Therefore, it is essential for service providers to ensure that their services are accessible and usable by a variety number of service requesters (Elia *et al.*, 2014).

4.8 Working Nature

Web services work usually in two request/response operation modes: asynchronous and synchronous. In the asynchronous mode, a web service requester does not expect to get a response back directly. So, the time in this mode is not crucial. Therefore, a requester will issue a request to perform some tasks and a provider is not expecting to send back a required response instantly. In other words, the response will be available when the provider decides. In the synchronous mode, a service requester will be waiting for getting back a response after issuing a request. Therefore, the requester is expecting to get the response back directly and the provider has to handle it instantly (Mannava & Ramesh, 2012).

4.9 Infrastructure Control

Usually, web services reside in more than one foreign infrastructure. Besides, they require a tight integration process between both environments of a service provider and a service consumer. This integration process depends totally on the network connectivity to work. Any fault in a provider's side has a direct effect on the process of integration and even on a consumer's environment. In many cases, faults lead to consumer's system crash (Cook & Barfield, 2006).

4.10 Release Control and Updating

Web service providers control everything regarding releases of their services. Generally, service releases are provided to make changes and recovers from minor or major issues. Sometimes changes in the body of a service are not evident from its Application Programming Interfaces (APIs). In other words, the changes do not require API reconstruction. As a result, service consumers do not notice such internal changes. However, as service APIs are the only visible parts to service consumers, any reconstruction in them affect directly the process of a service calling (Li *et al.*, 2013). It is worth mentioning that the API reconstruction process may include modification in the parameter-list, return type, and service name. Such modifications force consumers to make changes in their environments and to upgrade the calling process. Otherwise, they will face many calling issues or even they will not be able to call the required services at all.

4.11 Composition and Reusability

Composition in web services means connecting together various independent services with different functionalities to provide a high-level and more value-added functionality to the compositors (service-based applications or systems). Generally, composition can be classified into two types (Mathkour *et al.*, 2012) as follows:

- **Static composition:** It occurs in the compilation time. In this type, service requesters should specify all the required information of invoking required services while coding. The main drawback here is that composition details are hard-coded in the development phase and cannot be changed in runtime.
- **Dynamic composition:** It occurs in the runtime. In this type, service requesters should be able to discover methods of required services in runtime. The main drawback here is that the increased level of complexity due to searching and discovering of services.

Integrating many existing systems developed using different programming languages, deployed on different platforms, at different locations, and makes them work together as a collaborative system. Web services come to the picture to solve this issue using a number of standards. For example, web services use XML for data exchanging among different systems regardless of their languages, platforms, and locations. Regarding reusability, using web services prevent the repetition of developing existing applications and programming tasks. Therefore, web services are the best choice for integrating different existing systems. Besides, using this technique saves a lot of time, cost, and effort.

4.12 Complexity

Developing web services is more complex than web applications due to a number of reasons. For example, the development process involves using external entities such as UDDI brokers for registering services. Besides, the development process uses heavy methods for communicating and exchanging data including SOAP and XML. All these reasons and many others mentioned in previous sections require a lot of deployment, registering, and integration processes.

5. CONTRIBUTIONS AND CONCLUSIONS

5.1 Contributions

After studying both web applications and web services in detail, the authors proposed a set of requirements to help developers or anyone approaches this field of study to make the right choice when using these techniques to deliver the functionality of Internet-based software components. Table 1 presents briefly the comparison between both techniques

corresponding to a set of factors that have been derived from the proposed requirements given in this paper.

Table 1: Comparison between web applications and web services

Technique / Factors	Web applications	Web services
Orientation	Human-oriented	Machine-oriented
Accessibility	Accessed by web browsers	Accessed by services, applications, and systems
Development Tools	Developed using browser-oriented programming, scripting, and styling languages/frameworks alongside server ones	Developed using standard programming languages
Development Process	Easy	Difficult somewhat
Usability	Easy	Complicated somewhat
User Interfaces	Provided	Not provided
Structure View	Centralized	Distributed
Interaction Models	Not provided	Provided
Building Blocks	Presentation, business, and data access	SOAP, WSDL, and UDDI
Testing	Page death, linkrot, redirection, and validation	WSDL and PFB
Interoperability	Fully	Partially
Operation Modes	Synchronous	Synchronous and asynchronous
Infrastructure Control	Not integrated	Integrated
Updating	Users are not involved	Clients are involved
Composition	Not supported	Supported
Reusability	Not supported	Supported
Complexity	Low	High

5.2 CONCLUSIONS

This study aimed to clarify many aspects regarding using web applications and web services. Generally, they can fulfil the requirements proposed in this paper but in detail there are many differences between them. However, the paper is not to draw any conclusion regarding which technique is superior since the suitability of each technique is greatly influenced in one way or another by the application/service domain and usage. Choosing one technique to deliver the functionality of an Internet-based software component means that many requirements have to be considered carefully. The authors provided most of the requirements for web developers to go for the right choice in this context. For future works, the authors will consider some experimental scenarios and case studies to compare web applications and web services practically.

REFERENCES

Anass, M. & Ahmed, E. (2017). Towards a standard Resful WADL implementation of Multi-view web services. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(4), 315-320.

Cook, W. & Barfield, J. (2006). Web Services versus Distributed Objects: A Case Study of Performance and Interface Design. *IEEE International Conference on Web Services (ICWS)*, 419-426.

Danesh, A. S., Saybani, M. R., Yahya, S., & Danesh, S. (2011). Software release management challenges in industry : An exploratory study. *African Journal of Business Management (AJBM)*, 5(20), 8050-8056.

Doğan, S., Betin-Can, A., & Garousi, V. (2014). Web application testing: A systematic literature review. *Journal of Systems and Software*, 91(1), 174-201.

Elia, I., Laranjeiro, N., & Vieira, M. (2014). A Field Perspective on the Interoperability of Web Services. *IEEE International Conference on Services Computing (SCC)*, 75-82.

Fernandes, N., Costa, D., Duarte, C., & Carriço, L. (2012). Evaluating the Accessibility of Web Applications. *The 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-Exclusion (DSAI)*, 14, 28-35.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems (FGCS)*, 29(7), 1645-1660.

Khan, R., Khan, S., Zaheer, R., & Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. *The 10th International Conference on Frontiers of Information Technology (FIT)*, 257-260.

Lam, M. (2011). Methodologies, tools, and techniques in practice for Web application development. *Journal of Technology Research (JTR)*, 3, 1-20.

Lee, S. (2014). A Study on Web Service Analysis and Bio-information based Web Service Security Mechanism. *International Journal of Security and Its Applications (IJSIA)*, 8(2), 77-86.

Li, J., Xiong, Y., Liu, X., & Zhang, L. (2013). How Does Web Service API Evolution Affect Clients?. *The IEEE 20th International Conference on Web Services (ICWS)*, 300-307.

Lorenzo, G., Fasolino, A., Melcarne, L., Tramontana, P., & Vittorini, V. (2007). Turning Web Applications into Web Services by Wrapping Techniques. *The 14th Working Conference on Reverse Engineering (WCRE)*, 199-208.

Lucca, G. & Fasolino A. (2006). Testing Web Applications: The State of Art and the Future Trends. *Information and Software Technology (IST)*, 48(12), 1172-1186.

Mannava, V. & Ramesh, T. (2012). An Adaptive Design Pattern for Invocation of Synchronous and Asynchronous Web Services in Autonomic Computing Systems. *Asia-Pacific Web Conference (APWeb): Web Technologies and Applications*, 131-142.

Marinho, E., Mendonca, A., Rodrigues, G., Alves, V., & Bonifcio, R. (2011). Exploring Architecture-Based Reliability Analysis of Current Multi-layered Web Applications. *The 5th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, 51-60.

Mašovi, S., Sara, M., Kamberovi, H., & Kudumovi, M. (2012). Java technology in the design and implementation of web applications. *Technics Technologies Education Management (TTEM)*, 7(3), 504-514.

Mathkour, H., Gannouni, S., & Beraka, M. (2012). Web service composition: Models and Approaches. *International Conference on Multimedia Computing and Systems (ICMCS)*, 718-723.

Maximilien, E. & Singh, M. (2005). Toward Web services interaction styles. *IEEE International Conference on Services Computing (SCC)*, 1, 147-154.

Moreno, N. & Vallecillo, A. (2005). Modelling interactions between Web applications and third-party systems. *The 5th International Workshop on Web Oriented Software Technologies (IWWOST)*, 441-452.

Repasi, T. (2009). Software testing - State of the art and current research challenges. *The 5th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 47-50.

Sharma, A., Hellmann, T., & Maurer, F. (2012). Testing of Web Services - A Systematic Mapping. *The IEEE 8th World Congress on Services*, 346-352.

Tesarik, J., Dolezal, L., & Kollmann, C. (2008). User interface design practices in simple single page web applications. *The 1st International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, 223-228.

Torchiano, M., Ricca, F., & Marchetto, A. (2009). Defect Location In Traditional Vs. Web Applications - An Empirical Investigation. *The IEEE 11th International Symposium on Web Systems Evolution (WSE)*, 121-129.

Yu, Y., Lu, J., Fernandez-Ramil, J., & Yuan, P. (2007). Comparing Web Services with other Software Components. *IEEE International Conference on Web Services (ICWS)*, 388-397.