Sindh Univ. Res. Jour. (Sci. Ser.) Vol.50(003) 345-354 (2018)

http://doi.org/10.26692/sujo/2018.09.0059

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Context-aware Heterogeneous Service Composition Framework for Pervasive Computing Environments

M. ALJAWARNEH[++], L. D. DHOMEJA*, Y. A. MALKANI**

Institute of Information and Communication Technology, University of Sindh, Jamshoro, Pakistan

Received 20th February 2018 and Revised 03rd July 2018

Abstract: Pervasive computing offers environments in which user needs or tasks are fulfilled without demanding their attention. This requires discovering a service or a set of services and interacting with them in response to context (i.e. user presence, user activity, user location, temperature level, light intensity level etc.). In pervasive computing environments, services available in the environment may be heterogeneous with regard to different service discovery protocols (e.g., UPnP, SLP, JINI, etc.) being used for their publication, discovery and interaction. Contextual service composition may involve discovery of and interaction with heterogeneous services based on context, raising an issue of service heterogeneity in pervasive computing environments. In this paper, we put forward a framework that addresses this issue. The proposed framework has been designed, implemented and evaluated, details of which are provided in the paper. The performance evaluation results indicate the system can perform well in both local and distributed settings.

Keywords: Service composition, Contextual Service Composition, Service Discovery Protocols, Heterogeneous Services, Contextual Service Discovery, and Event-Condition-Action Policy.

1. INTRODUCTION

In 1991, Mark Weiser outlined the vision of the computers for the twenty-first century with the name of ubiquitous computing, now also known as pervasive computing. According to Mark Weiser's vision, the computing will move beyond desktops, become ubiquitous and invisible to the users (Weiser, 1991). Putting the light on Weiser's vision, (Satyanarayanan, 2001) elaborates the invisibility as "complete disappearance of pervasive computing technology from a user's consciousness" and relates this to "minimal user distractions". Achieving invisibility in pervasive computing requires allowing applications to adapt themselves in response to context (i.e. user presence, user activity, user location, temperature level, light intensity level etc.). This makes context-awareness in general and contextual service composition in particular, a core requirement for pervasive computing environments

Pervasive computing environments may have a number of devices (such as PDAs, smart mobile devices, smart digital TV receivers, smart display screens, smart cameras and smart electronic appliances) offering different services. In order for pervasive computing environments to be able to fulfil user needs without demanding their attention, services available in the environment need to be discovered and interacted with in response to context. The needs of the users may sometimes be met with a provision of an atomic service, but most of the times this may involve a composition of multiple services. The process of discovering services and composing them based on context is called contextual service composition. For example, a user may want to answer the phone call while moving around in the home. This requires that the speaker and micro-phone services be discovered and interacted with based on the user context (i.e. location). As an another example, a user has comeback home from work and wants to relax, and this may involve dimming light value of the room the user is sitting in (e.g. 40% light intensity value), moderating temperature of room (e.g., 20-degree room temperature), starting her favourite music (e.g. Jazzmusic). This requires discovering various services (e.g. light service, temperature controlling service, music service, speaker service etc.), composing them to meet the high-level user goal of relaxing in the room.

Service Oriented Architecture (SOA) is one of the promising approaches used for development of pervasive application because of its loosely-coupled architecture. Its loosely-coupled architecture allows the services be developed and managed independently of each other, which can be discovered and composed together to meet user needs in pervasive computing environment. A number of service discovery protocols have been developed which are based on SOA, such

[++] Corresponding Author's E-mail: maljawarneh@scholars.usindh.edu.pk
* Institute of Information and Communication Technology, University of Sindh, Jamshoro, Pakistan.
** Institute of Mathematics and Computer Science, University of Sindh, Jamshoro, Pakistan.

as JINI (Waldo, 2000), UPnP (UPnP, 2018), OSGi (Tavares and Valente, 2008), SLP (Guttman, 1999), Bluetooth SDP (Bluetooth, 2018), Bonjour (Bonjour, 2018), etc. These protocols provide mechanisms and supporting infrastructure for three primary tasks: (1) expression and publication of services, (2) discovery of services and (3) interaction with discovered services. While these protocols provide the support for three basic aforementioned tasks, they highly vary in mechanisms / techniques used to support these three tasks, thereby raising an issue of heterogeneity. In pervasive computing environments, an umber of devices are assumed to be available with each one offering one service or multiple services and these services may be based on different service discovery protocols, i.e., some of them may be UPnP-based, while others SLP-based and others JINI-based. An issue of heterogeneity arises in contextual service composition when the services involved in composition are built using different service discovery protocols. Supporting heterogeneous contextual services composition increases the potential and diversity of possible application that can be developed. One of the main goals of our research has been a provision of the framework for development of pervasive computing applications involving service composition based on context. One of the main goals of our research has been a provision of the framework for development of pervasive computing applications involving service composition based on context. One of the main objectives to achieving this goal is resolving an issue of heterogeneity of services. We address these issues and propose a framework for heterogeneous service composition in pervasive computing environments.

The reminder of this paper is as follows. Section 2 provides the background to service composition the description of related work on heterogeneous service composition. In section 3, we present our proposed heterogeneous context-aware service composition framework. Section 4 presents the performance evaluation of the proposed framework. Finally, the paper is concluded in section 5.

2. LITERATURE REVIEW

In this section we provide a brief background to service composition including approaches generally used in the literature to addressing the issue of heterogeneous service composition. This section also provides description of various research efforts focusing on heterogeneous service composition.

2.1 Background

Service Oriented Architecture (SOA) is the paradigm that facilitates a process of a service composition as services are built independently and maintained separately where they could later be used to dynamically build an application from those concrete services(composite service). SOA consists of mainly three parts **(Fig. 1)** (1) service provider which is responsible to build and publish the service, (2) a service directory that is responsible to register the services and answer the discovery messages from the service consumer and (3) service consumer that discovers the service and directly interacts with the service.

2.1.1 Service Composition Phases

The process of composition of the services goes generally into three phases:

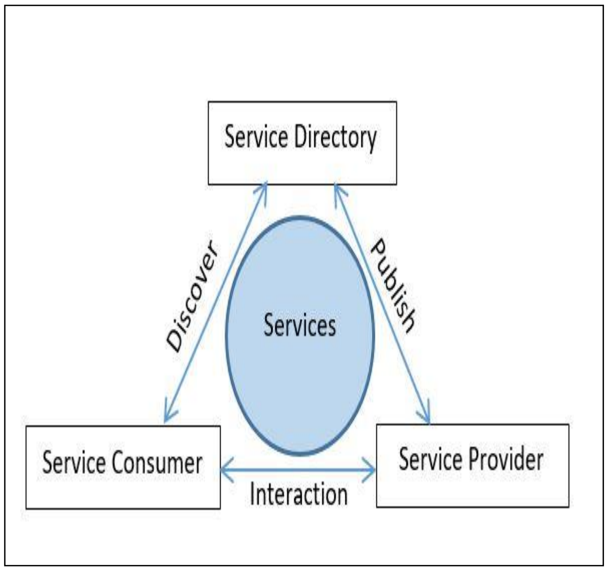


Fig 1: SOA Three Parts

a) Composition phase: this phase deals with generating the description of the composed service, which is usually written in a standard format understandable by the underlying supporting infrastructure. This file description contains the description of the individual services that are involved in the composition. Different languages have been developed to write the description of the composed services, such as Flow (Casati and Shan, 2001), BPEL (Khalaf, *et al.*, 2005), UML-WSC (Khalaf, *et al.*, 2002), etc. This description can later be used to trigger the composition of the services automatically by its corresponding execution middleware.

b) Selection phase: in this phase, the composed service is built by selecting concrete services from the available services (involved in the composition) based on the information available in the composed service description. In this phase, the services are discovered and bound. There are usually two approaches which are followed in this phase. (1) static approach and (2) dynamic approach. In the former, the services are known and selected at the design time, while in the latter;the services are unknown at the design time and are automatically discovered, selected and bound at runtime. In both cases, the outcome of the selection phase is an executable composed service.

c) Execution phase: this is the last phase of service composition where the executable composed service is executed by the underlying system by invoking and monitoring messages on the constituent services of the composed service.

2.1.3 Heterogeneous Service Composition

The issue of service heterogeneity arises when the participating services in the composed service are developed with different service protocols. In the literature, three different approaches have been proposed and used to address the service heterogeneity issue.

a) Direct (Transparent) Approach:

In this approach, service heterogeneity is addressed by translating the services from one service discovery protocol to another and registering them as logical services. This approach uses a software component (often called an adapter) for translation of a service from one protocol to another. The main problem with such approach is that the number of adapters increases with the number of services. The research efforts using this approach include (Allard, *et al.*, 2003; Yérom- *et al.* 2005; Delphinanto *et al.*, 2007; El Kaed, *et al.* 2011; Meliones, *et al.* 2010).

b) Intermediate (Explicit) Approach:

In this approach, service heterogeneity is addressed by translating the heterogeneous services (developed using different SDPs) to a common format, and then the common format is used for discovery of services. Realization of this approach includes two software components – one component to translate the service description to a common format and the second component to translate service invocation messages back to the protocol format. The research efforts using this approach include (Benmokhtar, *et al.*, 2008; Yerom-David *et al.*2006; Cheng, *et al.* 2012; El Kaed *et al.*2011; Kim et al., 2012; Koponen and Virtanen, 2004; Limam *et al.* 2007; Nakazawa, To *et al.*, 2006; Raverdy, *et al.* 2006; Yang, *et al.*2012)

c) Client Side Interoperability:

With this approach, the heterogeneity is achieved by making the client capable of interacting with different service discovery protocols. The research efforts using this approach include *et al.* 2011).

While all of three techniques have been used in the literature to address service heterogeneity issue, the literature suggests that intermediate (explicit) is more promising and widely used approach to addressing service heterogeneity in service composition, since it requires translating the heterogeneous servicesto a common format, allowing discovery of and interaction with the services in a unified way. In the following section, we provide a brief overview of systems exploiting intermediate approach to heterogeneous service composition, with a special focus on mechanisms used for expressing composed services and common platform used forconverting heterogeneous services into.

2.2 Related Work

(Álamo, *et al.* 2010) provide a framework that allows composition of heterogeneous services using a BPEL composition file. The service heterogeneity issue is addressed by translating all the services into a common platform based on OSGi.

(Chauvel, *et al.*2011) provide an approach to dynamic service composition of heterogeneous service-oriented systems. The service heterogeneity is addressed through translating the services written in a particular platform (e.g., UPnP, OSGi, etc.) into a corresponding web service (web service proxy). This means there would be a web service proxy for each of the protocol-specific service. The BPEL is used for the composed service description.

(Reyes, 2010) provide a service composition language called simple service composition language (SSCL) for composing heterogeneous services by extending OSGi framework. The heterogeneity issue is resolved using two proxies. That enables the discovery and interaction with the heterogeneous services.

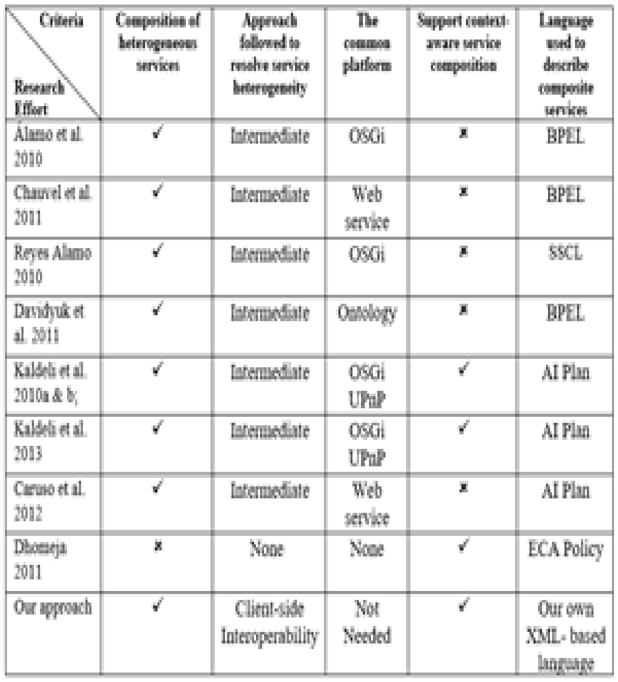
(Davidyuk, *et al.* 2011) provide the middleware called MEDUSA, The proposed middleware exploits Amli framework (Georgantas *et al.*, 2010) for discovery of heterogeneous services and ubiSOAP framework (Caporuscio *et al.*, 2009) for interaction with heterogeneous services. The users are provided with a set of interfaces to build a composite service.

(Kaldeli, *et al* .2010a) have developed a middleware supporting dynamic service composition of heterogeneous services. This middleware uses AI planning techniques for creating a plan for the composed services and rule engine for executing the created plan in response to events generated by the sensors in the environment. The service heterogeneity issue is addressed by translating protocol-specific services (e.g., UPnP, Web services) into OSGi services.

(Kaldeli, *et al.*., 2013) provide an extended version of their middleware discussed in (Kaldeli e*t al.*, 2010b). The added functionality includes (1) triggering the composition process (includes both planning and execution) in response to explicitly expressed user needs and (2) unlike their earlier work where each service was translated into OSGi format. All devices/services and their functionality are mapped to OSGi-UPnP standard format.

(Caruso *et al.*, 2012) provide a framework that uses two composition engines to enable service composition: the offline engine used for static service composition and the online engine for dynamic service composition with the help of AI planner that plans a composite service based on user needs. The developed framework wraps the non-SOAP based services and registers them in the repository as web services**.(Table 1)** shown below summaries the related work.

Table 1: Summary of Related Work



Criteria / Research Effort	Composition of heterogeneous services	Approach followed to resolve service heterogeneity	The common platform	Support context-aware service composition	Language used to describe composite services
Alamo et al. 2010	✓	Intermediate	OSGi	✗	BPEL
Chauvel et al. 2011	✓	Intermediate	Web service	✗	BPEL
Reyes Alamo 2010	✓	Intermediate	OSGi	✗	SSCL
Davidyuk et al. 2011	✓	Intermediate	Ontology	✗	BPEL
Kaldeli et al. 2010a & b;	✓	Intermediate	OSGi UPnP	✓	AI Plan
Kaldeli et al. 2013	✓	Intermediate	OSGi UPnP	✓	AI Plan
Caruso et al. 2012	✓	Intermediate	Web service	✗	AI Plan
Dhomeja 2011	✗	None	None	✓	ECA Policy
Our approach	✓	Client-side Interoperability	Not Needed	✓	Our own XML- based language

3. <u>PROPOSED CONTEXT-AWARE HETEROGENEOUS SERVICE COMPOSITION FRAMEWORK</u>

In order to address an issue of service heterogeneity in context-aware service composition, we have designed and implemented a framework called context-aware heterogeneous service composition framework. In the following, we present a high-level architecture of the framework along with a brief discussion of each of its components and also briefly discuss implementation of the framework.

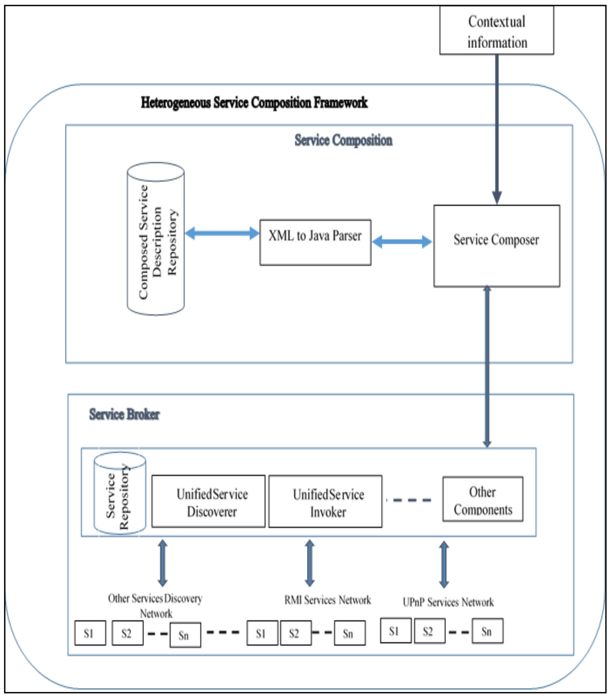


Fig 2: High-level Architecture of the Proposed Framework

3.1 An Architecture of the Proposed Framework

The high-level architectureof the proposed framework providing the support for heterogeneous service composition is shown in **(Fig 2).**

The proposed Framework has two main components: (1) Service Composition, (2) Service Broker. In the following subsection we discussed them in detail.

3.1.1 Service Composition

That is responsible to carry out the composition process as dictated in the composed service descriptions. The subsections presented next describe the responsibility of each subcomponent in a context-aware service composition process.

A. Composed Services Description Repository

This component is responsible for maintaining the description of the composed services in XML format file. The composed services may be developed by a developer. The XML file description will have the description of the constituent services of the composed service along with their default value of parameters associated with each of the constituent services.

B. XML to Java Parser

This component is responsible for parsing the XML description of the composed service and creating a Java object based on the parsed information. The created Java object will contain all required information for the constituent services and this object is handed over to the service composer component, which creates a composite service.

C. Service Composer

This component is responsible for creating an object of the composite service from the Java object containing description of the constituent services of the composite service. The creation of the composite service involves discovering the services based on context and binding them to itself, and then interacting with the bound services for performing actions using default parameter values of the services. The discovery and interaction involved in the process of creation of the composite service is done through the service broker component, which is discussed in the next section.

3.1.2 Service Broker

As we discussed above, the services involved in composition may be heterogeneous (i.e. from different discovery protocols). The service broker is responsible to hide the heterogeneity of these services by allowing the discovery of heterogeneous services and interaction with them in a unified manner, without the need for knowing the protocol used to develop these services. In the following subsections we discuss the service broker components.

A. Unified Service Discoverer

This component is responsible to allow the discovery of the services in their respective platform (the service discovery protocol used to develop the service), the broker allow the service composition to interact with heterogeneous services in a unified manner. The unified service discoverer discovers the services in two steps: (1) by discovering the protocol in which the services are developed and(2) initiating a service discovery request using that protocol.

B. Unified Service Invoker

This component allows the services to be invoked in a unified manner in the same way as the previous component by two steps: (1) finding the protocol used to develop the service, and (2) invoking the service using that protocol, without the need for a specialized component (e.g. proxy) to translate the invocation.

C. Service Repository

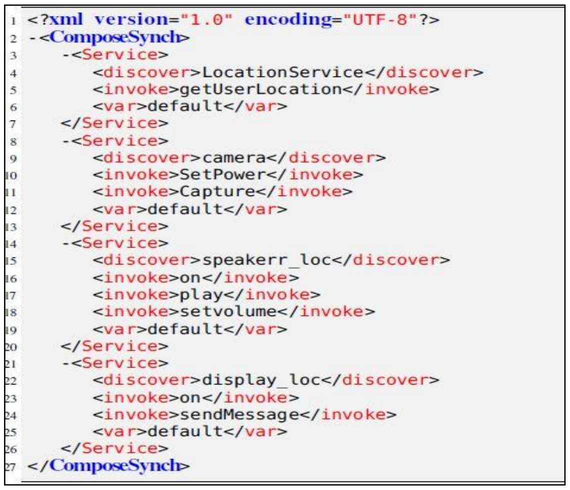
This component is responsible for receiving presence notification of each services and maintaining this information in the repository to facilitate the task of quick discovery of requested services. As soon as the service leaves the environment, its information is removed from the repository. While this is similar to what is supported by most of the discovery protocols, there are some discovery protocols that don't have this mechanism available such as RMI, JINI. Therefore, in case of RMI or JINI services, the discovered RMI and JINI services are cached in this component.

3.2Implementation

The proposed framework was implemented using Java programming language. The implementation of the proposed framework consists of various Java objects including service composer, XML to Java parser and service broker. These components interact with each other to realize heterogeneous service composition.

Through the following example scenario, we discuss implementation of the proposed heterogonous service composition framework.The example scenario is that: *"While the user at home, someone is knocking at the door by pressing the doorbell. As the doorbell rings, the camera mounted at the top of the main gate start capturing the person picture, and the image of the person is displayed at rendering device nearest to the location of the user at home (e.g. room1) and also the doorbell notification is played on the sound system nearest to location of the user at home".*

As can be noted from the example above, there are four services needed to be composed to realize the scenario: (1) camera service, (2) location service, (3) display service and (4)speaker service. When someone knocks the doorbell, the system will respond by discovering these services and composing them to meet the high level user need. The composition is context-aware in that it uses the location service to locate the current user location and discover the rest of the involved services based on that location. The XML description of notification composite service is shown in **(Fig 3).**



```
<?xml version="1.0" encoding="UTF-8"?>
- <ComposeSynch>
   - <Service>
       <discover>LocationService</discover>
       <invoke>getUserLocation</invoke>
       <var>default</var>
   </Service>
   - <Service>
       <discover>camera</discover>
       <invoke>SetPower</invoke>
       <invoke>Capture</invoke>
       <var>default</var>
   </Service>
   - <Service>
       <discover>speakerr_loc</discover>
       <invoke>on</invoke>
       <invoke>play</invoke>
       <invoke>setvolume</invoke>
       <var>default</var>
   </Service>
   - <Service>
       <discover>display_loc</discover>
       <invoke>on</invoke>
       <invoke>sendMessage</invoke>
       <var>default</var>
   </Service>
</ComposeSynch>
```

Fig 3: XML description of notification composite service

In order to run this scenario on the implemented system we have developed simulated context widget to simulate the doorbell notification context in order to trigger the composition of the services as dictated by the composite service description. The GUI that allow sending the name of the composite service is shown in **(Fig 4).**

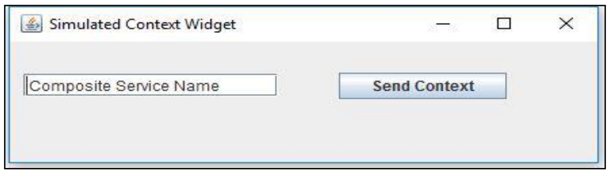


Fig 4: Simulated Context Widget

When the *Send Context* button is pressed, the simulated context widget passes to service composer the name of the composite service, which is the name of the composed service description file saved in the repository. The message sequence diagram as illustrated below **(Fig 5)** shows interaction involved among various components of the framework for realizing heterogeneous service composition in response to context.

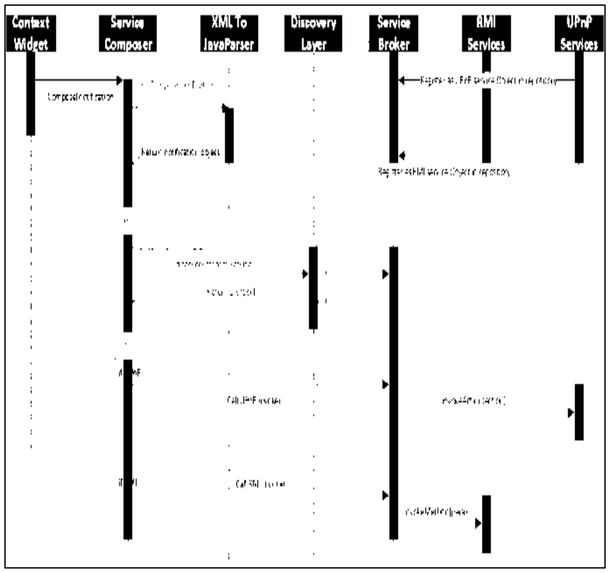


Fig. 5: Message Sequence Diagram for Heterogeneous Service Composition

To summarize, the proposed framework provides the support for composing heterogeneous service. The service heterogeneity issue within the framework is addressed by the service broker.

Unlike traditional approaches to service heterogeneity where each service developed using a particular protocol (e.g., UPnP, SLP) is translated to a common platform, our framework allows discovery and interaction with heterogeneous services in a unified manner, hence eliminates the overhead associated with translating and managing each service to the common platform.

4. PERFORMANCE EVALUATION

We present the performance of the proposed framework for heterogeneous contextual service composition. Tests conducted under both local and distributed environment settings include the service composition time. The service composition time is the time taken by the system for reading the composed service description, building the service composition object and firing the composed service object in response to context.

We run the scenario discussed earlier 20 times in both local and distributed settings and measure the total composition time. The composition time starts from a point the composite service name is sent by the context widget until the default values on all services involved in the composition are implemented.

4.1 Local Testing Environment Setup

In the local setting, we tested the system by running all system components on one machine with this configuration: Intel core i5 3360M 2.8 processor with 4GB RAM, windows 10 64bit operating system with jdk1.7. In this local setting, the proposed system (heterogeneous service composition framework), the services from different protocols such as UPnP, RMI, etc. and the context widget used to send the composite service name were running on the same machine. The local setting configuration is shown in **(Fig 6)**.

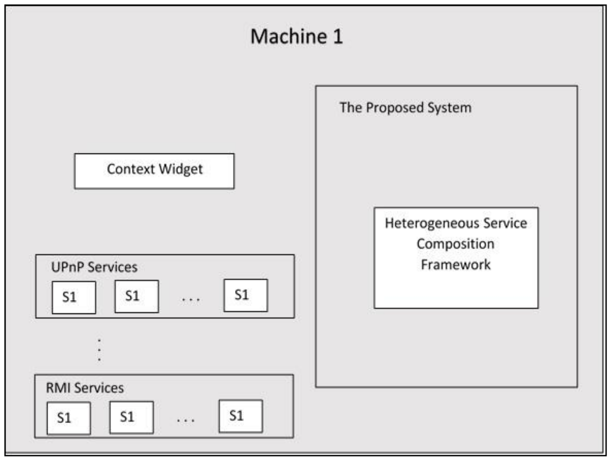


Fig 6: Running the System in Local Settings

Test 1: Service composition time in a local setting.

Results: The reported composition time for the test 1 are the average times and presented graphically in **(Fig. 7)** along with the standard deviation. The fig 7 shows the average total composition time of 69ms with

7.85 standard deviations. The total composition time is divided into three parts: (1) reading the composed service description from the XML file saved in the repository, (2) building the composed service object, this involve discovering the services and binding them to the composed service objectand (3) interacting with constituent services by implementing the default values on them.

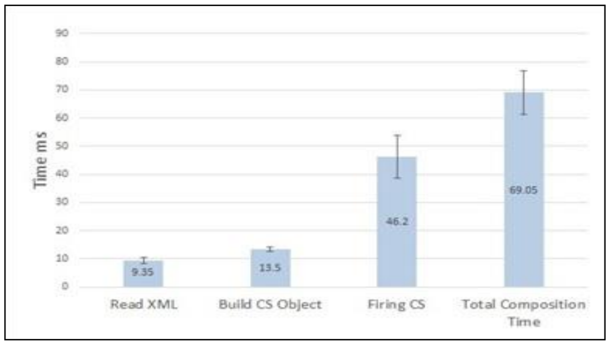


Fig 7: Average Total Composition Times in Local Setting

The fig 7 clearly indicates that the time for discovering the services is 13.5ms, which involves the discovery of services from two heterogeneous protocols. Unlike existing approaches in which the service description from every protocol is translated intoa common format, in our approach the services are discovered in their native platform, thus eliminating the overhead associated with translation. The fig 7 indicates an average time taken by the system to interact with the discovered services for implementing the user preferences is 46.2ms. This time includes the time taken by the system for interaction with two services from UPnP and two services from RMI protocol in their native protocols.

4.2 Distributed Testing Environment

In distributed setting, three machines were used. Machine 1 had these specifications: Intel core i5 3360M 2.8GHz processor, 4GB RAM with Wi-Fi IEEE 802.11 network card, running windows 10 64bit operating system with jdk1.7. Machine 2 had theses specifications: Intel core2duo E5300 2.6GHz processor, 2GB RAM with Wi-Fi IEEE 802.11 network card, running windows XP 32bit operating system with jdk1.7. Machine 3 had these specifications: Intel core i3 2350M 2.3GHz processor, 4GB RAM with Wi-Fi IEEE 802.11network card, running windows 10 64bit operating system with jdk1.7.

Machine 1 was used to run the proposed heterogeneous service composition framework and the context widget. Machine 2 was used torun RMI services and RMI registry, while machine 3 was used to run the UPnP services. All these machines were connected together on the same network using Wi-Fi access point of Institute of Information and Communication Technology (IICT) at University of Sindh to form a distributed testing environment. The distributed setting configuration is illustrated in **(Fig. 8)**.

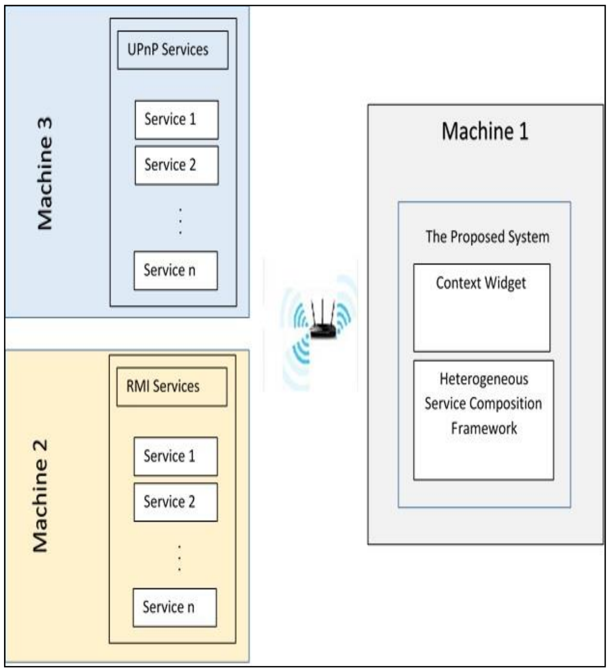


Fig 8: Running the System in Local Settings

We ran the same test(discussed earlier)in distributed setting 20 times (fig8).

Test 2: Service composition time in a distributed setting.

Results: The reported composition time in distributed setting for the test 2 are the average times and are presented graphically in fig 9 along with the standard deviation. The **(Fig. 9)** shows the average total composition time of 158ms with 13.42 standard deviations.

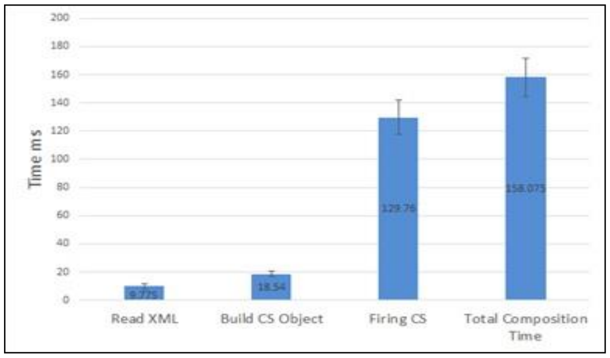


Fig 9: Average Total Composition Times in Distributed Setting

As it can be noted from the fig 8 and 9, total composition time in local settings is 69ms, while in distributed settings it is 158ms. This increase of 89ms is due to the network overhead associated with the distribution of the services across the network. The major contribution in the total composition time in distributed settings is of the time taken by the system for remote interaction with the services to implement user preferences, which is around 130ms.

5. CONCLUSION

One of the core challenges of pervasive computing environments is a provision of a customized service to best meet user needs. The provision of such a customized service may lead to a process in which the services are discovered and composed based on contextual information (such as user activity, user location information, nearby resources, etc.). This makes contextual service composition a central requirement for pervasive computing applications. In contextual service composition, an issue of service heterogeneity may arise when the services required in service composition are heterogeneous (for example, some may be JINI-based, while others UPnP-based, etc.). In this paper we have addressed this issue and proposed and implemented the framework called Context-aware Heterogeneous Service Composition Framework for Pervasive Computing Environments. This paper has provided a detailed description of an approach used in the framework to address an issue of service heterogeneity, the architecture of the framework along with its components, implementation and performance evaluation. This paper also reviews some relevant systems and summarizes them in the table mentioning what approach has been taken by them to address the service heterogeneity issue. Other core research challenges that need to be addressed in the field of context-aware service composition in pervasive computing may well include (1) decoupling of adaptation decision logic from other parts of an application to achieve simplification of development and dynamic programmability and (2) user involvement.

However, the proposed framework can further be extended to support fault tolerance to cope with service and power failures – by developing a specific language for service composition, where the recovery plan of failure could be incorporated with the composed service description. Currently, the user preferences are manually included in the composed service description, this can be further improved by developing a separate component for user preferences known as user profile and dynamically including the user preferences from her profile in the composed service description.

REFERENCES:

Álamo, J. M. R., H. I. Yang, J. Wong, and C. K. Chang, (2010). Automatic service composition with heterogeneous service-oriented architectures. In *Aging Friendly Technology for Health and Independence* 9–16. Springer.

Allard, J., V. Chinta, S. Gundala, and G. G. Richard (2003). Jini meets UPnP: an architecture for Jini/UPnP interoperability. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on* 268–275.

Benmokhtar, S., P G. Raverdy, A. Urbieta, and R. S. Cardoso, (2008). Interoperable semantic and syntactic service matching for ambient computing environments. In *1st International Workshop on Ad-hoc Ambient Computing (AdhocAmC).*

Bluetooth, S. I. G. (2018). Bluetooth specification, in "http://www.bluetooth.org/en-us/specification" Last accessed 20,8,2015. *Bluetooth Specification.*

Bonjour, A. (2018). Apple Bonjour, in "https://www.apple.com/support/bonjour/" Last accessed 20,8,2015. *Apple Bonjour.*

Bromberg, Y. D. and V. Issarny, (2005). INDISS: Interoperable discovery system for networked services. In *Proceedings of the ACM/IFIP/USENIX 2005 international Conference on Middleware* 164–183.

Bromberg, Y. D., V. Issarny, and P. G. Raverdy, (2006). Interoperability of service discovery protocols: Transparent versus explicit approaches. In *the 15th IST Mobile and Wireless Communications Summit.*

Caporuscio, M., P. Raverdy, H. Moungla, M. Caporuscio, P. Raverdy, and H. Moungla, (2009). ubiSOAP : A Service Oriented Middleware for Seamless Networking Issarny To cite this version : ubi SOAP : A Service Oriented Middleware.

Caruso, M., C. Di Ciccio, E. Iacomussi, E., Kaldeli, A. Lazovik, and M Mecella,. (2012). Service ecologies for home/building automation. In *Proc. 10th International IFAC Symposium on Robot Control (SYROCO).*

Casati, F., and M. C. Shan, (2001). Dynamic and adaptive composition of e-services. *Information Systems*, *26*(3), 143–163.

Chauvel, F., G. Hu, and L. Mei, (2011). Dynamic interoperability between heterogeneous services. In

Proceedings of the 2011 international workshop on Networking and object memories for the internet of things 7–8.

Cheng, S. T., C. H. Wang, and G. J. Horng, (2012). OSGi-based smart home architecture for heterogeneous network. *Expert Systems with Applications*, *39*(16), 12418–12429.

Davidyuk, O., N. Georgantas, V. Issarny, and J. Riekki, (2011). MEDUSA: Middleware for end-user composition of ubiquitous applications. *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, *11*, 197–219.

Delphinanto, A., J. J. Lukkien, A. M. J., Koonen, F. T. H., den Hartog, A., Madureira, I., Niemegeers, and F.Selgert, (2007). Architecture of a bi-directional Bluetooth-UPnP proxy. In *Proceedings of the 4th Annual IEEE Consumer Communications and Networking Conference, CCNC 2007, 11-13 January 2007, Las Vegas, NV, USA, 34-38.*

El Kaed, C., Y. Denneulin, and F. G. Ottogalli, (2011). Dynamic service adaptation for plug and play device interoperability. In *Proceedings of the 7th International Conference on Network and Services Management*p. 46–55.

Georgantas, N., V. Issarny, S. Mokhtar, Ben, S. Bianco, G. Thomson, and P. Raverdy, (2010). *Handbook of Ambient Intelligence and Smart Environments*. https://doi.org/10.1007/978-0-387-93808-0

Grace, P., G. S. Blair, and S. Samuel, (2003). ReMMoC: A reflective middleware to support mobile client interoperability. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE* 1170–1187. Springer.

Guttman, E. (1999). Service location protocol: Automatic discovery of IP network services. *Internet Computing, IEEE*, *3*(4), 71–80.

Kaldeli, E., E. U. Warriach, J. Bresser, A., Lazovik, and M. Aiello, (2010a). Integrating, composing and simulating services at home. In *Int. Conf. on Service Oriented Computing (ICSOC).*

Kaldeli, E., E. U. Warriach, J. Bresser, A. Lazovik, and M. Aiello, (2010b). Interoperation, composition and simulation of services at home. In *Service-oriented computing* 167–181. Springer.

Kaldeli, E., E. U. Warriach, A., Lazovik, and M Aiello,. (2013). Coordinating the web of services for a smart home. *ACM Transactions on the Web (TWEB)*, *7*(2), 10.

Khalaf, R., Mukhi, N., Curbera, F., and Weerawarana, S. (2005). The business process execution language for web services. *Process-Aware Information Systems*, 317.

Kim, J. E., Boulos, G., Yackovich, J., Barth, T., Beckel, C., and Mosse, D. (2012). Seamless integration of heterogeneous devices and access control in smart homes. In *Intelligent Environments (IE), 2012 8th International Conference on* 206–213.

Koponen, T., and Virtanen, T. (2004). A service discovery: A service broker approach. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (p. 7--pp).

Limam, N., Ziembicki, J., Ahmed, R., Iraqi, Y., Li, D. T., Boutaba, R., and Cuervo, F. (2007). OSDA: Open service discovery architecture for efficient cross-domain service provisioning. *Computer Communications*, *30*(3), 546–563.

Meliones, A., Economou, D., and Liverezas, I. (2010). Network adaptation in intelligent environments. In *Intelligent Environments (IE), 2010 Sixth International Conference on* (pp. 225–230).

Nakazawa, J., Tokuda, H., Edwards, W. K., and Ramachandran, U. (2006). A bridging framework for universal interoperability in pervasive systems. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* 3Pp.

Park, H., B. Kim, Y., Ko, and D. Lee, (2011). InterX: A service interoperability gateway for heterogeneous smart objects. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on* 233–238.

Raverdy, P. G., V. Issarny, R. Chibout, and A. de La Chapelle, (2006). A multi-protocol approach to service discovery and access in pervasive environments. In *Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference on* 1–9.

Reyes A J. M. (2010). *A Framework for Safe Composition of Heterogeneous Soa Services in a Pervasive Computing Environment with Resource Constraints*. Iowa State University, Ames, IA, USA.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, *8*(4), 10–17.

Tavares, A. L. C., and M. T. Valente, (2008). A Gentle Introduction to OSGi. *SIGSOFT Softw. Eng. Notes*, *33*(5), 8:1--8:5. https://doi.org/10.1145/1402521.1402526

Thöne, S., R. Depke, and G. Engels, (2002). Process-oriented, flexible composition of web services with UML. In *ER (Workshops)* (pp. 390–401).

UPnP. (2018). UPnP Fourm. In *About the UPnP Plug adn Play Forum," in http://www. upnp. org" last accessed 20,8,2015*.

Waldo, J. (2000). *The Jini Specifications*. (Arnold, Ed.) (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*. https://doi.org/10.1038/scientificamerican0991-94

Yang, H.-I., R. Babbitt, J. Wong, and C. K. Chang, (2012). A framework for service morphing and heterogeneous service discovery in smart environments. In *Impact Analysis of Solutions for Chronic Disease Prevention and Management* 9–17. Springer.