IJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 5 Issue 1 January, 2017

The Distributed Students' Portal Framework

Mohammed Awad[1]; Rami Raba[2]

Faculty of Information Technology, University of Palestine [1]; College of Intermediate Studies, Al-Azhar University – Gaza[2]

m.awad@up.edu.ps[1]; ramiraba@yahoo.com[2]

ABSTRACT

This paper describes a newly defined Distributed Students' Portal Theoretical Framework in which all academic and administrative departments of a university portal are distributed to components of a complete portal. These components are software packages or modules that encapsulate a set of related data. The users (students or staff) that are connected to the internet/intranet can view and update their profiles and specific needs via registration and admission, students' affairs, financial affairs, graduate studies, library, and/or other extended components. Service Oriented Architecture (SOA) is used for structuring the new framework.

Keywords: Students' portal, SOA, distributed components, loose coupling, tight coupling.

1. INTRODUCTION

As shown in figure 1, and as a part of SOA structure, the Orchestration Point is the station to verify the validity of user and to identify his/her request.

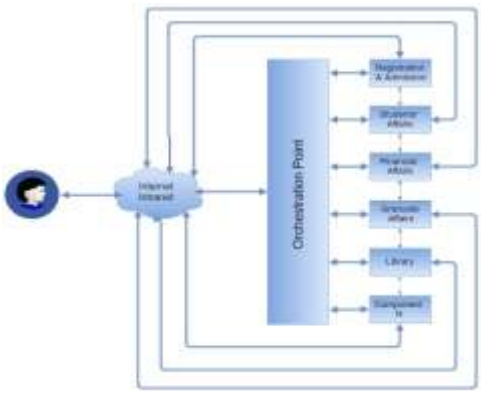

Fig 1: The New Students' Portal Components Framework

The new students' portal framework that is based on SOA has several advantages over the traditional framework. These advantages are summarized in table 1 that shows a comparison between the traditional framework and the new frameworks regarding to coupling, distribution, interoperability, maintainability, cost, complexity, and reusability.

As shown in table 1, the traditional framework components are tightly-coupled. This traditional framework shows that all portal components are dependents on each other. On the other hand, the new framework is a loosely-coupled framework with independent components. In addition, the distribution of the traditional portal framework faces difficulties to distribute the portal components in an orchestrated way that can easily be implemented in university systems. A change in one component in the traditional framework leads to other changes in other components. However, the new framework completely supports components distribution and interoperability. It shows that portal components of a university are distributed portal components, and the framework allows any component to be added or modified without any effect on other components. Furthermore, the traditional framework components have limited reusability. The traditional framework shows that any portal component might be more difficult to reuse because dependent components must be included. On the other hand, the new framework components are reusable in which any portal component can be reused because dependent components are already included. Moreover, the traditional framework components have impediments to maintain, in which portal components might require more effort and/or time to maintain due to the increased inter-component dependency. However, the new

framework components are easy to maintain due to the loose coupling features that isolate components and then isolate components maintenance. The traditional portal framework is difficult to extend. Any change or extension on the traditional framework usually forces a ripple effect of changes to other components because of lack of distribution. On the other hand, the new portal framework components are easily extensible, which shows that any portal component of a university can be extended or modified without any effect on other components. In addition, the traditional framework components are more complex, in which portal components depend on each other's components, which increases time and effort of components coding, while the new framework components are less complex because portal components are isolated components, which does not require more effort to code the portal components.

The loose-coupling features of the new portal framework together with other features shown in table 1 reduce the cost of implementing and maintaining a university portal.

Table 1. Comparison between traditional framework and new framework

	Traditional Framework	New Framework
Coupling	Components are Tightly-Coupled [1]	Components are Loosely-coupled
Distribution	Difficult to Distribute [2]	Supported
Interoperability	Not Supported [3]	Supported
Maintainability	Has Impediments [4]	Easy to Maintain
Extensibility	Difficult to Extend [5]	Easy Extensible
Cost	Relatively High [6]	Relatively Low

For consolidating the new framework, the specifications of the framework are explored in the following sections.

2. ORCHESTRATION POINT SPECIFICATIONS

Since the restructuring process of the new students' portal framework is highly dependent on SOA and based on SOA standards [7, 8], it is compulsory to orchestrate the services available in the students' portal by creating an orchestration point.

It establishes a common point of integration for other components or applications, which enables an implemented orchestration point to be the main integration point. In addition, the orchestration point leads to an increase in the students' portal flexibility because of the following reasons:

- The workflow logic encapsulated by an orchestration can be modified or extended in a central location [9].
- Positioning an orchestration centrally can significantly ease the merging of students' portal processes by abstracting the interfaces that connect the corresponding automation solutions together [9].

By establishing a service-oriented integration for students' portal architecture, orchestration can support the evolution of students' portal framework, because it is the main factor of any successful students' portal tool based on the new students' portal framework, which contains various components based on different computing platforms. In addition, the orchestration is the heart of SOA-based students' portal because it establishes means of centralizing and controlling a great deal of inter and intra-students' portal logic through a standardized service model. Furthermore, it expresses a body of business process logic that is typically owned by a single student's portal tool. TYPESET TEXT

3. SPECIFICATIONS OF EXTENDING THE NEW STUDENTS' PORTAL FRAMEWORK

Sometimes, there is a need to add an additional component to the basic students' portal framework to meet special business needs. In this case, the new students' portal framework provides the extensibility feature to make it possible and simple to extend the framework to suit these special business needs. Then, if it is needed to add a new component to the basic students' portal framework, there are few steps to be followed. These are concluded from the these references [10-14].

a) Prepare a distributable module for the extra component.

b) Design and develop the business logic of the extra component.

c) Encapsulate the extra component in a Web service.

d) Register the Web service in the orchestration point as a partner service.

e) Reconfigure the orchestration configuration files to suit the new extra component.

f) Reconfigure the composite students' portal application to involve the new extra component.

g) Redeploy the whole stuff and test the application again to assure that other components are not affected by the extra one.

4. META-MODEL FOR THE NEW DISTRIBUTED STUDENTS' PORTAL THEORETICAL FRAMEWORK

As a contribution to the body of knowledge, this section explores the novelty of this research work by providing a meta-model that describes the design of the framework in a standard way. The new students' portal framework meta-model is designed in this section to support and represent the new students' portal framework and its concepts. A modeling language can be either a graphical or textual language [15, 16]

For the purpose of designing the new student portal framework, Unified Modeling Language (UML) is used as the graphical modeling language [17-19], UML is formally defined by a meta-model (or semantic model) and it is used to represent software design since it is widely used for modeling both research and industry works [17, 20-22]. UML provides notations for specifying the packaging of a logical design into components that represent a distributed computing architecture and this can be modeled using a UML component diagram [17, 18, 22].

Component diagram supported by other UML notations is used to create a meta-model for the new framework. The components of the meta-model and the relationships among these components are explored in this section. Figure 2 represents a meta-model for the new framework.

The new student's portal framework is composed of main components. These are: student's portal and orchestration point.

The student portal component uses the orchestration point component as a dynamic look-up mechanism to provide information about student's portal services. The orchestration point component thus enables optimized access to service meta-data and management of service interactions and policies. In its most elemental state, it is composed of service meta-data artifacts documents, such as XML Schema Definition language (XSD) or Web Services Description Language (WSDL) files. These service meta-data files are managed by the Orchestration Point component.

The student portal component represents the connectivity of the new students' portal frameworks components, and it aggregates with student portal provider Bus and student portal requester Bus components to control the flow of messages and instructions between the student portal requester and the student portal provider.

In addition, it provides support to communication protocols used for the communication among the distributed student portal framework components. The student portal component using message Model, Instructions flow, and communication protocols components. The message model component enables the student portal to support different types of message models flowing between the provider and requestor.

The communication protocols component provides support for different communication protocols, such as SOAP and HTTP to connect the providers with the consumers (the communication protocols support several interactions patterns, such as request/response, publish/subscribe, and synchronous/ a synchronous). The instructions flow component contains interfaces to invoke the mediation flows to perform mediation between student portal requestor and student portal provider, and to provide references to external services.

The student portal component has the student portal message Router component routes requests to specific business services based on defined criteria. It is conditional routing, and it usually requires the selection of some piece of data from the message upon which routing can be based. In addition,

student portal Message Router component optionally uses service engine component to provide business logic, as well as consume student portal services.

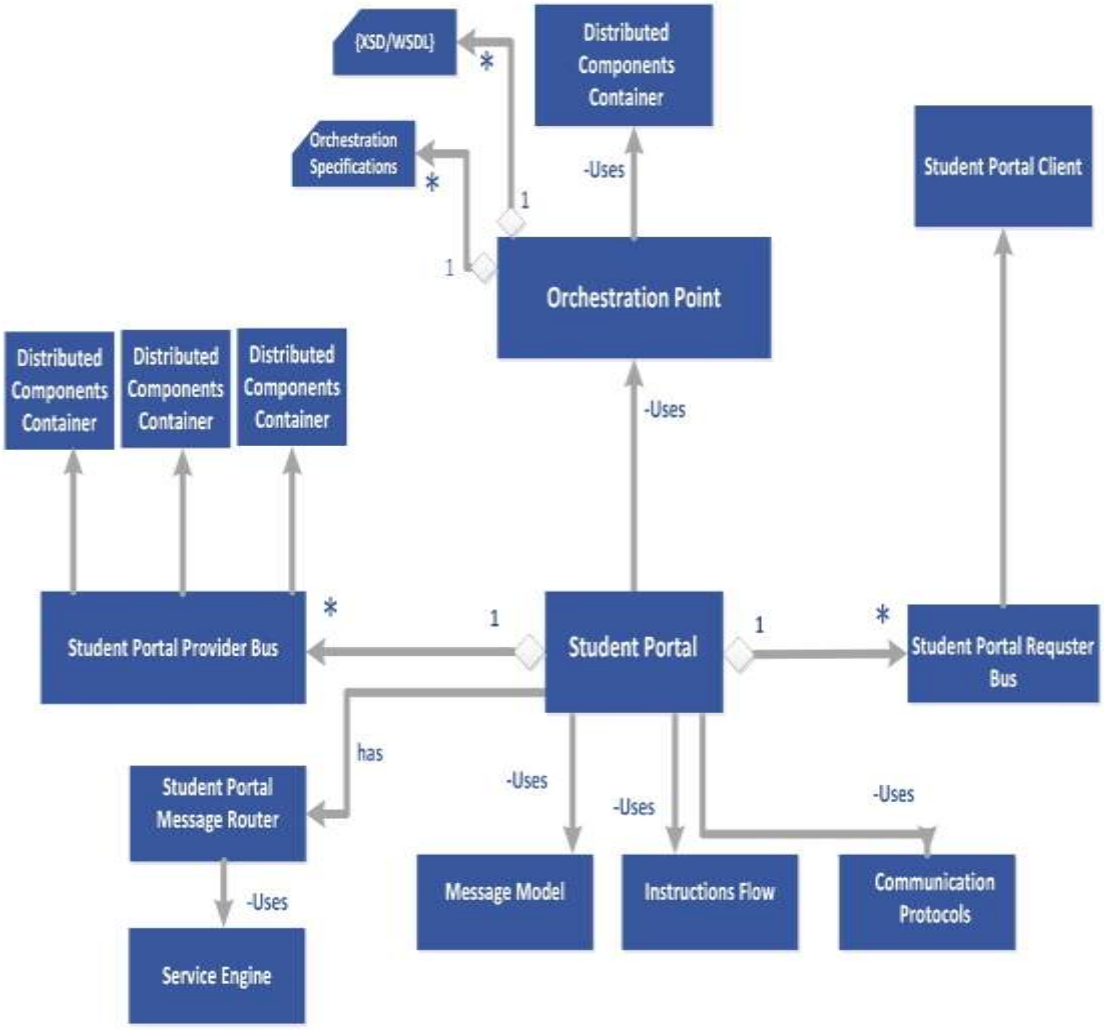

Fig 2: Meta-Model for the new framework

The Orchestration Point component and student portal distributed components use Distributed component containers as run environments.

The container in its general definition is the interface between component functionalities and the application server functionalities that support the component [23].

5. TEXTUAL MODELING OF THE DISTRIBUTED STUDENTS' PORTAL THEORETICAL FRAMEWORK

For consolidating the Meta-Model and supporting the novelty of this research work, Extended Backus-Naur Form (EBNF) is used as a textual modeling language to support the understanding of the graphical meta-model by following its flexible features and symbols of describing meta-models textually [16, 24].

The specific EBNF definitions for the main components of the meta-model are presented and then followed by the complete EBNF definitions for the meta-model. This section presents the specific EBNF definitions for the main components of the meta-model. Table 2 presents the meaning of some EBNF symbols that are included in the EBNF definitions of the Distributed Students' Portal framework.

Table 2. EBNF symbols summary [24, 25]

Symbol	Description
::=	The element to the left of the symbol is defined by the constructs on the right.
*	The preceding construct may occur zero or more times.
{...}	The constructs within the curly braces are grouped together.
[...]	The constructs within the square brackets are optional.
|	An exclusive OR.

The Distributed Students' Portal framework is composed of a number of distributed components. If these components are called: Distributed Component 1, Distributed Component 2, Distributed Component 3, (optional) Distributed Component 4, and Orchestration point. Then the EBNF definition of this part is as follows:

Students' Portal ::= Distributed_Component_1 ,Distributed_Component_2, Distributed_Component_3, [Distributed_Component_4],Orechestration_Point, Students_Portal_Service_Bus;

The distributed components are described by the following EBNF definitions, as follows:

Distributed_Component_1 ::=
Component1_Queries_and_operations,(Distributed_Component_Container | User_Distribution_Settings);

Distributed_Component_2 ::=
Component2_Queries_and_operations,(Distributed_Component_Container | User_Distribution_Settings);

Distributed_Component_3 ::=
Component3_Queries_and_operations,(Distributed_Component_Container | User_Distribution_Settings);

Distributed_Component_4 ::=
[Component4_Queries_and_operations], (Distributed_Component_Container | User_Distribution_Settings);

In addition, the syntax of Orchestration Point component is specified by the following EBNF definitions:

Orchestration_Point ::=
Orchestration_Specifications,{Orchestration_Specifications}, XSD_WSDL, {XSD_WSDL},(Distributed_Component_Container | User_Distribution_Settings);

The Distributed Students' Portal framework components exchange data and messages through an enterprise service bus. An enterprise service bus represents an environment designed to foster sophisticated interconnectivity between services. It establishes an intermediate layer of processing that can help overcome common problems associated with reliability, scalability, and communications disparity [26, 27]. The Students' Portal components of the new Students' Portal framework use the application server service bus to interchange messages and data among Students' Portal components. This service bus is mentioned as Students_Portal_Service_Bus in the EBNF description of this meta-model.

The Students' Portal Service Bus component uses the Orchestration Point component as a dynamic look-up mechanism to provide information about Students' Portal services. The Orchestration Point component thus enables optimized access to service meta-data and management of service interactions and policies. It also supports the integration of other standard registries and repositories. In its most elemental state, it is composed of service meta-data artifacts documents, such as XML Schema Definition Language (XSD) or Web Services Description Language (WSDL) files. These service meta-data files are managed by the Orchestration Point component. The syntax of The Students' Portal Service Bus component and other components that have direct relations with Students' Portal Service Bus component are specified by the following EBNF definitions:

Students_Portal_Service_Bus ::=
{Students_Portal_Service_Provider_Bus | Students_Portal_Service_Requester_Bus}, Students_Portal_MessageRoute, Message_Model , {Message_Model}, Instructions_Flow, Communication_Protocol, {Communication_Protocol}, Orchestration_Point;

Students_Portal_Service_Provider_Bus ::=
(Distributed_Component_1 | Distributed_Component_2 | Distributed_Component_3 |

[Distributed_Component_4] |
{Distributed_Component_4});

Students_Portal_Service_Requester_Bus ::=
Students_Portal_Client_Web_Service,
{Students_Portal_Client_Web_Service};

The Students' Portal Service Bus component represents the connectivity of the Distributed Students' Portal framework components, and it aggregates with Students' Portal Service Provider Bus and Students' Portal Service Requester Bus components to control the flow of messages and instructions between the Students' Portal Service Requester and the Students' Portal Service Provider. In addition, it provides support to communication protocols used for the communication among the Distributed Students' Portal framework components. The Students' Portal Service Bus component uses Message Models, Instructions Flow, and Communication Protocols components. The Message Models component enables the Students' Portal Service Bus to support different types of message models flowing between the service provider and service requestor. The syntax of Communication Protocols component is specified by the following EBNF definition:

Communication_Protocol ::= (SOAP | HTTP),
(Request_Response | Publish_Subscribe |
Synchronous_Asynchronous);

The Communication Protocols component provides support for different communication protocols, such as SOAP and HTTP to connect the service providers with the service consumers (The communication protocols support several interactions patterns, such as request/response, publish/subscribe, and synchronous/asynchronous.). The Instructions flow component contains interfaces to invoke the mediation flows to perform mediation between Students' Portal service requestor and Students' Portal service provider, and to provide references to external services.

The syntax of Students' Portal Message Router component is specified by the following EBNF definitions:

Students_Portal_MessageRouter ::=
[Service_Engine],
{Students_Portal_Client_Request,
Input_Student_Service, Response_Redirect};

The Students' Portal Service Bus component has the Students' Portal Message Router component routes requests to specific business services based on defined criteria. It is conditional routing, and it usually requires the extraction of some piece of data from the message upon which routing can be based.

For realizing the framework of this research, a prototype was developed. The analysis, design, development, and testing, of the SOA-based students' portal prototype was done based on the specifications and the meta-model of the framework presented in this paper.

Two types of testing are done for the purpose of verifying that the prototype meets the framework design and specifications. A web interface combined all the components of the prototype in one composite application as an interface for the user to do testing and execution of any students' portal web service. The types of testing carried out for the purpose of validating the prototype are unit testing, Web services testing, and compatibility testing.

Moreover, a case study was conducted for the purpose of evaluating the prototype of this research that validates the new students' portal framework in real environment. One organization was selected to conduct the case study required for evaluating the research prototype. The selected organization is Al-azhar University-Gaza in Palestine. The organization is selected because it has a students' portal to accomplish specific business needs.

Al-azhar University-Gaza database system was chosen because it has eight departments, registration and admission, academic affairs, personal affairs, student affairs, finance, library, and archive. All the mentioned departments databases aggregate and were loaded in a central DBMS. The students' portal needs the data from all departments stored in a single server that uses oracle 6i (tightly-coupled system). The data used to conduct this case study is generally textual data that describes the records related to the performance of the students in oracle i6 with its schema and format.

The University IT unit programmers face impediments when they want to maintain or extend any department related to the system. Therefore, they always prefer distributable functionalities of

software, which easily help them to grant privileges to the programmers.

Most of the data from the departments in Al-azhar University- Gaza serves the students through the students' portal. Students' portal components has to be independent, easily maintained, and extended, and these features are available in the SOA- based students' portal prototype.

After conducting the case study, and analyzing the responses of the IT unit team at Al-Azhar university, the observed results of conducting the case study proved that new framework has enabled the distribution and interoperability among the students' portal components.

6. CONCLUSION AND FUTURE WORK

This paper has explored the specifications for the conceptual distributed students' portal framework. In addition, a meta-model using both UML and EBNF is designed abstractly for consolidating the design of the distributed framework. This research has focused on how a conceptual framework for students' portals including the feature of component distribution be defined to enhance the current tightly-coupled portals, while it has not handled the other features of this portal, specifically, the academic features that any academic portal should have, and how these features are affected by an SOA architecture of the portal. Therefore, this could be a future research work that completes the optimum usage of an SOA based students' portal.

7. REFERENCES

[1]. Y. L. Mao Tan, "Design and Implementation of General Distributed Heterogeneous Data Exchange System," IEEE, 2011.

[2]. P. Wehrle, Miquel, M., &Tchounikine, A, "A Grid Services-Oriented Architecture for Efficient Operation of Distributed Data Warehouses on Globus," Proceedings of 21st International Conference on Advanced Networking and Applications, 2007.

[3]. J. X. Xiaohong Hu, Hongjie Liu, "Research of Architecture Pattern Based on .NET Distributed System," IEEE, 2011.

[4]. W. Zhang, "2-Tier Cloud Architecture with Maximized RIA and SimpleDB via Minimized REST," IEEE, 2010.

[5]. S. M. Rahul Ramachandran, Helen Conover and Chris Lynnes, "TALKOOT SOFTWARE APPLIANCE FOR COLLABORATIVE SCIENCE," IEEE, 2009.

[6]. J. Browne, "Securing a sustainable future for higher education: an independent review of higher education funding and student finance," 2010.

[7]. M. Barai, Binildas, & Caselli,V., "Service Oriented Archeticture with Java," UK: Packt Publishing, vol. 1 st ed., 2008.

[8]. M. Awad, Jebreen, I., & City, J., "Interoperable Distributed Data Warehouse Components," IJCSI, vol. 2, p. 275, 2015.

[9]. T. Lam, & Minsky, N., "Regulating Orchestration in SOA-Based Systems," Proceedings of 2010 Seventh International Conference on Information Technology, pp. 690-695, 2010.

[10]. D. S. n. Zhaleh Alimoradi, Dr. Hossein M. Shirazi, Dr.Alireza Khatoni, Narges Shahhoseini, "Design Portal Enterprise Services using Enterprise architecture methodology (Case Study: Portal Kermanshah University of Medical Sciences)," IEEE, pp. 435 - 439, 2010.

[11]. S. H.-j. Ni Dan, Chen Yuan, Guo Jia-hu, "Attribute Based Access Control (ABAC)-based cross-domain access control in service-oriented architecture (SOA)," IEEE, pp. 1405-1408, 2012.

[12]. A. Masood, "Cyber Security for Service Oriented Architectures in a Web 2.0 World: An Overview of SOAVulnerabilities in Financial Services," IEEE, pp. 1-6, 2013.

[13]. I. B. D. Yoseph Ismail Nurhasan, Hanif Fakhrurroja, "Information Model Design as Model-Driven for Service Oriented Architecture (SOA) Implementation in PK-BLU Institution Using SOA Ontology," 2013.

[14]. L. Al- Hakim, "Business Web Strategy: Design, Alignment, and Application," IGI, Global, pp. 135 - 136, 2008.

[15]. C. Kobryn, "Modeling Components and Framework with UML," Communications of the ACM, p. 43, 2000.

[16]. OMG, "Common Warehouse Metamodel (CWM) Specification," USA : OMG Headquarters., vol. 1st ed. Vol.1, 2003.

[17]. C. Kobryn, "Modeling Components and Framework with UML.," Communications of the ACM, p. 43 (10), 2000.

[18]. J. T. Lujanmora, "Physical Modeling of Data Werehouses Using UML.," ACM Journal., 2004.

[19]. N. Koch, & Kraus, A. , "The expressive power of uml-based web engineering," IWWOST02, vol. 16, 2002, June 2002.

[20]. W. K. Kruchten, Bran, & Slice, "Describing Software Architecture with UML.," Relational Software, 2001.

[21]. T. K. H. Atkinson, "Rearchitecting the UML Infrastructure.," ACM Transactions on Modeling and Simulation, vol. 12 (4), pp. 290-321, 2002.

[22]. OMG, "Unified Modeling Language.," from http://www.uml.org/, Retrieved 29/10/2011 2011.

[23]. E. Armstrong, Ball, J., Bodoff, S., Carson, D. B., Evans, I., Green, D, "The J2EE™ 1.4 Tutorial (2nd ed) " San Antonio Road Palo Alto, CA, USA: Sun Microsystems., 2004.

[24]. M. G. Yong Xia, "Rigorous EBNF-based Definition for a Graphic Modeling Language.," Winterthurerstr, 190,CH-8057 Zurich, Switzerland, 2002.

[25]. A. P. S. E. R. Gargantini, "Deriving a textual notation from a metamodel by University of Bergamo,"Available:https://doc.telin.nl/dsweb/Get/Rendition50041/3M4MDA_2006_online_proceedings.pdf#page=41, 1/12/2011 2007.

[26]. D. Salter and F. Jennings, Building SOA-Based Composite Applications Using NetBeans IDE 6, 1st ed. USA: PACKT Publishing, 2008.

[27]. S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More, 1st ed. NY, USA: Prentice Hall, 2005.

