**Available Online at www.ijcsmc.com/journal**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**
**IMPACT FACTOR: 7.056**

***IJCSMC, Vol. 11, Issue. 10, October 2022, pg.1 – 15***

# Exploratory Study to the Common Architecture of University Computerized Systems and the Solution to its Architectural Problems

**Mohammed M I Awad**
Faculty of Information Technology, University of Palestine, Palestine, m.awad@up.edu.ps

**DOI:** https://doi.org/10.47760/ijcsmc.2022.v11i10.001

***Abstract— This research has been focused on exploring the common architecture of university computerized systems (UCSs) to discover the common gaps and problems available in the architecture of most universities worldwide. Three universities have been selected to study and explore the common architecture of their systems; in addition to the literature of other universities. As a result, there are several problems in this architecture; these problems are direct results of tight-coupling, stand-alone applications, integration gap and interoperability gap. Therefore, this research has presented Service Oriented Architecture (SOA) as a solution to the mentioned problems. The solution has been described and a prototype has been designed, implemented and tested successfully as a proof to concept.***

***Keywords— University Systems Architecture, Distribution, Interoperability, SOA, Loose Coupling***

## I. INTRODUCTION

All universities worldwide must have computerized systems for their academic and administrative activities. UCSs improve the university collaborative activities as they are becoming the beating heart of universities as they integrate most of the resources and activities [1-3].

A university or any academic institution is an educational institution dedicated to education and research, which grants academic degrees in various disciplines. It operates with integrity in its financial, academic, personnel, and auxiliary functions. In addition, it establishes and follows policies and processes for fair and ethical behaviour on the part of its governing board, administration, faculty, and staff [4].

The great importance of UCSs at universities has given a motivation to do an exploratory research on the available systems at universities, how these systems work, and how they are built. On top of that, it is very essential to explore the structure of these systems and their dependences on each other's. The following section discusses this in details.

## II. COMMON ARCHITECTURE OF UCSS

An intensive exploratory research has been done to explore the common systems available at most of the universities. In addition, this exploratory research has focused on studying the dependences of those systems on each other's; as an introduction to study the structure of these systems and their architecture and how these systems interact with each other's. Three universities have been selected for conducting this exploratory

 

research. These are: Universiti Utara Malaysia [5], Limkokwing University of creative technology [6] and university of Palestine [7]. These three universities were selected because the researcher has worked at the three universities as a programming department head, a lecturer, an IT unit director and recently the vice president for the University of Palestine. After conducting this intensive research, a general conclusion of the systems and their dependences on each other's has been concluded. This conclusion is summarized and shown in figure 1, while table 1 explains the relations between those systems.

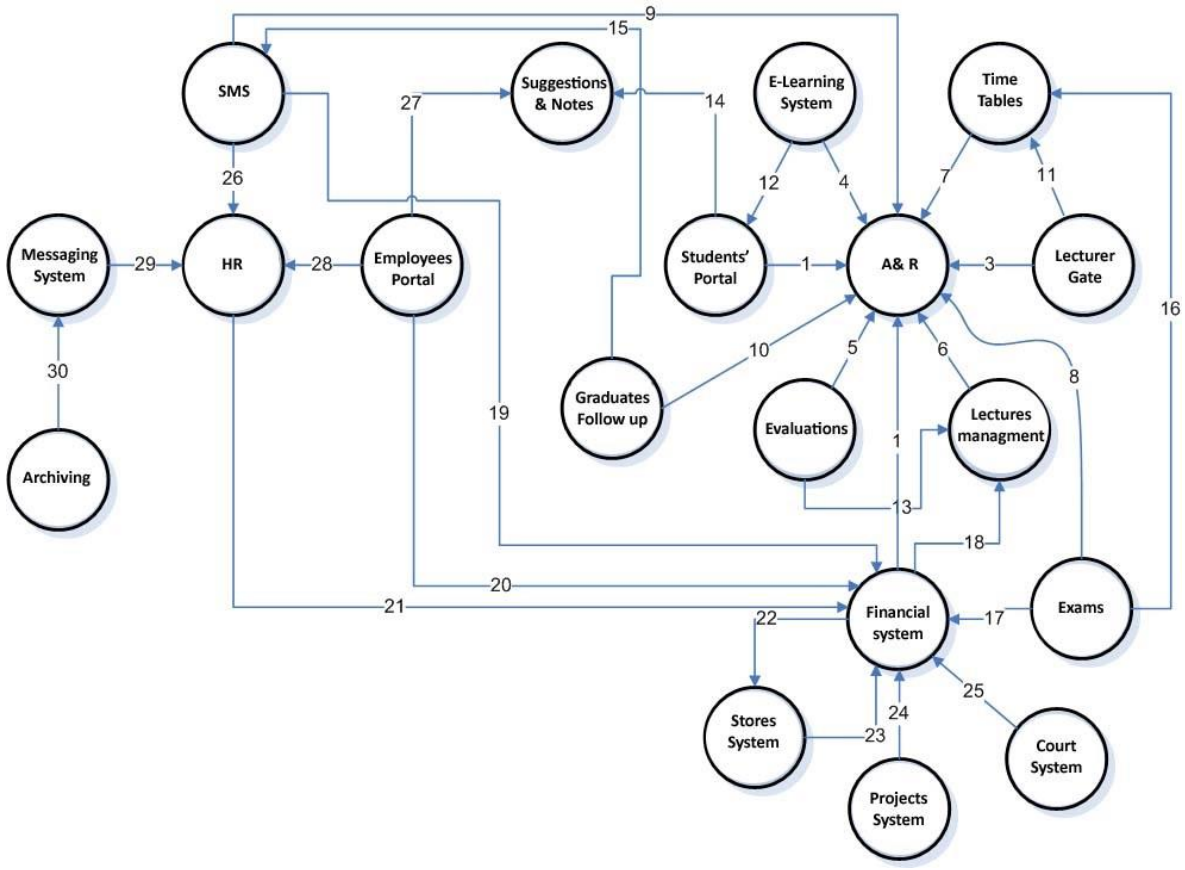


Fig. 1 Common architecture of UCSs as a conclusion of this exploratory research

TABLE I
DESCRIPTION TO THE ARCHITECTURE OF FIGURE 1

| Relation Number | Description |
|---|---|
| 1 | The students' portal is fully linked with the admission and registration (A & R) system, as it allows a student to follow up his academic record from the moment he registers at the university until his graduation; by knowing the courses offered for registration and the possibility of registering his courses according to his study plan according to the regulations and laws. |
| 2 | The admission and registration (A & R) system is linked with the financial system to synchronize the fees and all related payments; in addition to all other related financial transactions between the two systems. |
| 3 | The lecturer gate is linked with the admission and registration system, so that from the moment the teacher is approved to teach at the university, an account is created for him on the admission and registration system to follow up on his students, and extract academic transactions related to him/her. |
| 4 | The e-learning system obtains the data of the offered courses, students and teachers according to each semester from the admission and registration system and transfers it to the e-learning system to enable students and lecturers to interact with this system by downloading the knowledge material for students and the ability for the student to interact with it. |
| 5 | The evaluation system for evaluating faculty members depends in most of its data on the admission and registration system so that the system reads the lecturers' data and the courses offered to them from the admission and registration system |
| 6 | The lectures management system depends in its data on the admission and registration system mainly, where the academic performance of the lecturer is monitored in terms of his attendance and departure from |

 

| | |
|---|---|
| | lectures and the average of the total hours given, as well as follow-up his of e-learning records in terms of uploading his lectures to students and his interaction with the systems designated for that. |
| 7 | Time tables are extracted from the admission and registration system by obtaining the courses offered in each semester and distributing them electronically according to specific criteria and the available capabilities of lecturers and classrooms so that they can distribute the courses automatically according to working hours and allow the issuance of time tables. |
| 8 | The examination system depends on the admission and registration system so that it is fed with the names of the courses, lecturers and students who are allowed to deal with the system according to the proposed standards and conditions. The system also depends on the academic agenda to determine the dates allowed to activate the system, in addition, its reliability is to transfer the degree of the course submitted to the admission and registration system to be approved. |
| 9 | The SMS system depends on the admission and registration system, so that the messaging system feeds the students' data and phone numbers, classified according to their faculties. The system allows sending students' grades via their phones if the degree is finally approved, as well as informing them of any emergency. |
| 10 | The follow-up system for students and university graduates (employment) depends on the admission and registration system so that graduates' data are obtained according to their faculties in order to allow the university and the student to communicate through this system. |
| 11 | The lecturer gate depends partially on the time tables system, so that the time tables of lecturers are obtained through this system to follow up their lectures, as well as the schedules and dates of the proposed exams. |
| 12 | Through his portal login, the student uses the e-learning system so that s/he follows up on her/his lectures according to the registered courses and interacts with them by uploading lectures, handing in assignments, providing quizzes and other services available through the system. |
| 13 | The evaluation system relies on the lectures management system in terms of obtaining the lecturer's attendance data, his regularity in lectures, his use of e-learning systems (academic performance) and giving him a percentage within the lecturer's general evaluation. |
| 14 | The suggestions and notes system depends on the student portal, so that the system allows the student, through their portal; to make comments and suggestions through the forms designated for that, in addition, it allows feedback on data in this regard. |
| 15 | The graduates' follow-up system depends on the SMS system so that the graduates are contacted by sending short messages to inform them of the university's announcements, job opportunities and available trainings. |
| 16 | The exams system depends on the time tables system in terms of activating the exams system according to the examination dates in the time tables system, as well as closing the system. |
| 17 | The exams system depends on the financial system in terms of activating examinations for students who have paid their tuition fees within the standards and regulations. |
| 18 | The financial system depends on the system of lectures in terms of obtaining reports on the number of hours taught by the teacher to be financially approved. |
| 19 | The SMS system depends on the financial system in terms of sending messages to employees or students to follow up on their financial affairs. |
| 20 | The employee portal depends on the financial system, where the employee is able, through his personal account, to follow up his salary chip immediately after the final approval of the payroll and inform him of the details of his salary, including deductions, deductions, bonuses, and others. |
| 21 | The human resources system feeds the financial system with the employees' financial data in terms of the degree of residence, the beginning and end of their contract and their family data in order to issue payroll statements, determine bonuses and discounts, and all changes that have an impact on the salary. |
| 22 | The financial system feeds the stores system with the names of the items and the approximate prices for each item. |
| 23 | The stores system feeds the financial system with the required items and approximate prices for each item, as well as damaged and stagnant items for financial evaluation. |
| 24 | The projects system feeds the financial system with the names of projects, the financial value of each project, and all reports and other related reports in this regard. |
| 25 | The courts system feeds the financial system with all the cases and their financial value, and it feeds them with all the financial procedures for each case. |
| 26 | The SMS system depends partially on the human resources system so that the names of the employees, their phone numbers, and their administrative or academic classifications are obtained to inform them about urgent issues. |
| 27 | The notes and suggestions system relies on the employee portal, which allows the employee, through specific forms; to communicate with the concerned authority. |
| 28 | The employee portal depends entirely on the human resources system, so that the system provides the employee portal with all the information that pertains to the employee from the beginning of his appointment to the end of his services in terms of vacations, follow-up of his attendance and departure, follow-up of his punishment record, and also allows the employee to submit some requests through |

 

| | |
|---|---|
| | approved forms such as submitting vacations permissions through the gate. |
| 29 | The messaging system depends on the human resources system in terms of following the administrative ladder in force in the university, so that the human resources system feeds the messaging system with the administrative structure to be followed, and the messaging system depends on the data received from human resources in terms of the sending and receiving messages. |
| 30 | The archiving system depends on the messaging system, where the archiving program is fed with data through the messaging system that organizes the internal messaging process between departments and sections, classifying these messages according to incoming and outgoing and other classifications as needed. |

## III. GAPS AND PROBLEMS

The previous section has summarized the common university computerized systems, their relationships and dependences on each other's, in addition to describing what does every system do. Table 1 and figure 1 constitute the summary of the extensive exploratory research has been done on several universities (including University Utara Malaysia [5], Limkokwing University of Creative Technology [6] & University of Palestine [7]) to come out with these conclusions.

Common problems that are available at all researched universities explore that their systems are either stand-alone or tightly-coupled applications. These problems are direct results to several gaps noticed at these researched UCSs. These gaps are summarized by lack of: distribution, reusability, portability, extensibility and interoperability.

For bridging the mentioned gap and therefore solving the mentioned problems, the following section describes the solution in details.

## IV. SOA-BASED SOFTWARE ARCHITECTURE FOR EDUCATIONAL INSTITUTES

This section consolidates and further defines the solution of the research problems. Defining the new UCSs architecture consists of some mandatory specifications. These specifications are clear and strict enough to make sure that interoperability and distribution of the UCSs components are exist. In particular, UCSs vendors or developers who are interested to adopt the new UCSs architecture as a base to produce their own UCSs tools have to consider a set of specifications regarding: Distribution, Web services, SOA (Service Oriented Architecture) orchestration processes, XML schemas, WSDL documents, SOA-based composition, and the architecture extensibility. These specifications are discussed in this section. Figure 2 illustrates the new UCSs architecture.

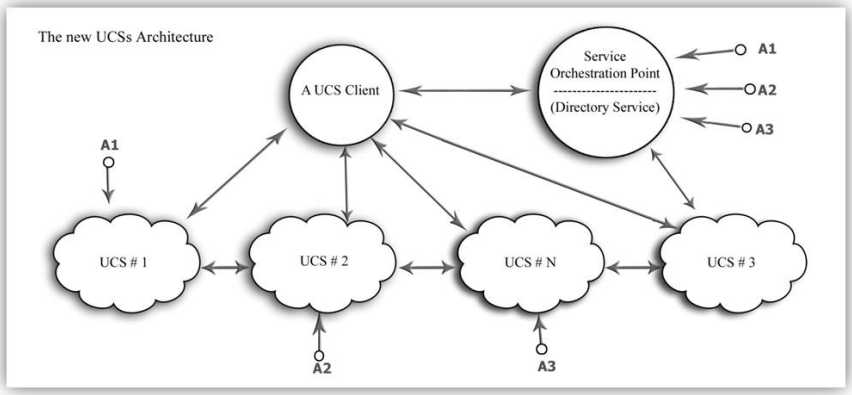


Fig. 2 The new UCSs architecture

Based on what is shown in figure 2, the details of the figure are described below.

- *Service Orchestration Point*: Describes the services available in its domain which are any systems, sub systems or components of computerized systems at a certain university. Those services are called service providers and register themselves in the orchestration point.
- *Service providers:* Each of them is a component that performs a service in response to a consumer request. UCS #1 to UCS # N are service providers.
- *Service consumers*: Each of them is a component that consumes the result of a service supplied by a provider: the main service consumer in the architecture is the client which represents UCSs administrators, also the service providers can be service consumers to other services.

 

- *Service interface:* Defines the programmatic access of the services, establishes the identity of the service and the rules of the service invocation.

The relationship between a service provider and consumer is dynamic and established at runtime by a binding mechanism done by the orchestration point. This dynamic binding minimizes the dependencies between the service consumer and service provider, which supports the loose coupling feature of the architecture [8, 9].

To clearly show the flow of actions when a client demand an execution of a UCS functionality, a simple flow diagram (that is adopted from these references [8-11] with customization to UCSs architecture )is shown in Figure 3.

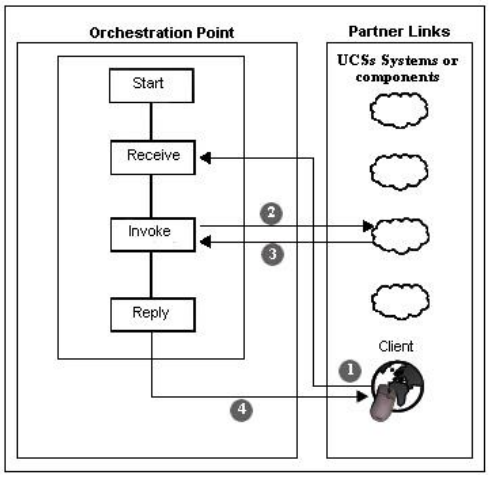


Fig. 3 Flow diagram for steps to consume a UCS service by a client

(1) When a client needs to execute a certain UCSs service, the orchestration point starts with a receive activity in which it receives the client request; (2) then, proceeds with invoking the suitable UCSs service; (3) the required service confirms the invocation request and (4) finishes by replying back to the client. An orchestration point process typically interacts with one or more UCSs web Services (the orchestration point process is also a web service). These UCSs web services are called also partner services or external service.

There are many distributed architectures that could be followed to meet the distribution specification of the new UCSs architecture. CORBA (Common Object Request Broker Architecture) [12], RMI (Remote Method Invocation) [13], RPC (Remote Procedure Call) [14, 15] and EJB (Enterprise Java Beans) [16] are samples of valid distribution architectures. Any of these distribution architectures or others could be followed to fulfil the distribution requirement of the UCSs architecture. By fulfilling this requirement, a distributed multi-tiered UCSs application can be accomplished. The application logic should be divided into components according to function, and the various UCSs components should be (optionally) installed on different machines; depending on the tier which the application component belongs to.

Java 2 Enterprise Edition (J2EE) or sometimes called JavaEE [17] is an open source architecture that supports the distribution concept and the multi-tiered application architecture. In addition, it provides all the tools might be needed to implement a distributed project including application servers, database connectors, development and deployment IDEs, and testing tools. Furthermore, J2EE is platform independent because it is built over Java language that is completely platform independent. Therefore, if the standards of EJB architecture (the distribution model of J2EE) is followed, the distribution shows two multi-tiered J2EE applications divided into these tiers:

- Client-tier components run on the client machine.
- Web-tier components run on web container of the J2EE server.
- Business-tier components run on the EJB container of the J2EE server.
- Enterprise information system (EIS)-tier runs on the EIS server.

The technology associated with the realization of SOA is Web services. A typical Web service (adopted from [14, 18]) needed in the new UCSs architecture is comprised of:

- *A decoupled technical service contract:* based on the standards of WSDL and XML Schema, the service contract has to consist of a WSDL definition and an XML schema definition. This service contract declares public functions (operations).
- *A body of programming logic*: based on web services standards, this logic may be custom-developed for the Web service, or it may exist as legacy logic that is wrapped by a Web service for its functionality to be made available via Web services communication standards. In the case that logic is custom-developed, it is created as components and is referred to as business logic.
- *Message processing logic:* based on web service communication standards, message processing logic exists as a combination of parsers and processors. Much of this logic is provided by the runtime

 

environment, but it can also be customized. The programs that carry out message-related processing are primarily event-driven and therefore can intercept a message subsequent to transmission or prior to receipt. It is common for multiple message processing programs to be invoked with every message exchange.

A Web service can be associated with temporary roles, depending on its utilization at runtime. For example, it acts as a service provider when it receives and responds to request messages, but can also do the role of service consumer when it is required to issue request messages to other Web services. For instance, a certain UCS can be a provider to the client and a consumer to the other system or component service in case two systems/components are executed in one batch.

To achieve the benefits of adopting SOA, based on SOA standards, every distributed component of the UCSs architecture has to be wrapped with a separated Web service. Each of these services is called an external service. External services (partners) and the client service interact with the orchestration point process. This process starts and ends somewhere, and involves the interaction of at least one external partner. In the UCSs architecture, the partner is the web service that has the ability to process the client's request, i.e., the UCSs components. The UCSs administrator sends his request for processing. Then, the orchestration point receives the request, invokes a web service that processes the request, and returns the response back to the UCSs administrator.

Remote Invocations of Web service operations of the UCSs architecture are asynchronous, i.e., the service provider must be capable of accepting requests from clients without notice. The designer of Web services needs to decide how to ensure that his or her implementation is compatible with the way in which a service provider supports asynchronous operations. To achieve this asynchronous feature, according to SOA standards, it is highly recommended that each Web service includes only one operation. The asynchronous scenarios for a UCS Web service would include:

- Production and transmission of a request message by a client.
- Consumption of the request message by the service provider.
- Production and transmission of a response message by the service provider.
- Consumption of the response message by the client.

These scenarios points have to be considered in any implementation of the UCSs architecture, and it is left to the UCSs designer to choose the technologies to achieve these.

In addition, there is a need for some standards to be considered when developing and integrating a UCS system/component based on the new UCSs architecture. The Web Services Definition Language (WSDL) provides a standard for describing services, the location of services, and what operations these services provide, all in a platform and language independent way. This allows the UCSs application developer to build loosely coupled services, which is one of the key factors of building successful SOA-based UCSs systems.

For specifying the format of WSDL documents that are supposed to be followed to implement UCSs Web services based on the new UCSs architecture, the WSDL file is divided into 5 main elements, all lie under the main <definitions> element:

- <types/>: Describes the data types used by the UCSs operations; described for the service. These data types are defined using XML Schema Definitions.
- <message/>: Describes the messages that are used within the UCSs service.
- <portType/>: Describes the operations that are available within the UCSs service.
- <binding/>: Describes what binding the service is using (Up to the time of writing this thesis it is SOAP)
- <service/>: Describes connection details for the specific bindings.

Moreover, since the restructuring process of the new UCSs architecture is highly dependent on SOA, therefore, based on SOA standards, it is compulsory to orchestrate the services available in the UCSs through 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 UCSs flexibility because:

- The workflow logic encapsulated by an orchestration can be modified or extended in a central location.
- Positioning an orchestration centrally can significantly ease the merging of UCSs processes by abstracting the interfaces that tie the corresponding automation solutions together.

By establishing a service-oriented integration for UCSs architecture, orchestration can support the evolution of UCSs architecture, because, it is the main factor of any successful UCSs system based on the new UCSs architecture that contains various components based on different computing platforms. In addition, it can be said that the orchestration is the heart of SOA-based UCSs because it establishes means of centralizing and controlling a great deal of inter and intra-UCSs logic through a standardized service model. Furthermore, it expresses a body of business process logic that is typically owned by a single UCSs tool. Furthermore, it establishes a business protocol that formally defines a business process definition.

The workflow logic within an orchestration is broken down into a series of basic and structured activities that can be organized into sequences and flows.

 

In creating the orchestration point, the whole creation process is up to the UCSs designer except some compulsory features which are:

- *Create reusable partner services*: make sure that partner services have to be reusable and can be used across different orchestration processes.
- *Asynchronous communication*: Coordinate asynchronous communication between different web services.
- *Data manipulation*: Data can be manipulated before exchanging between different services. The orchestration process can check, verify, and modify data from the client before sending requests to partner services, which simplifies the administration and troubleshooting of the UCSs.
- *Conditional and parallel processing is possible with orchestration services:* If system/component is already done, the orchestration process can send the request to another UCSs partner service, which supports the automation process of the UCSs.

As an important part of this section, based on SOA standards and after developing all the components of the UCSs; based on the new UCSs architecture, these components must be composed in one applications interface. This application has to be deployable as a composite UCSs tool ready to be executed and used by UCSs administrators (clients).

A composite application is an accepted solution that addresses a specific business problem by bringing together business logic and data sources from multiple systems. Typically, a composite application is associated with UCSs business processes, and may bring together several process steps, presenting them to the client (UCSs administrators) through a single interface that is customized to suit the requirements of the UCSs business needs.

SOA describes a category of composite applications composed of service provider and service consumer components that distributes business logic and offers location transparency for the service providers and consumers. The SOA approach allows the replacement or upgrading of individual components in the application without affecting other components [11, 19].

Figure 5.3 is the basic architecture specification for the way of composing the UCSs interface application. The numbers from (1) to (9) refer to the flow of actions done starting from (1) the client request to execute a UCS functionality and finishing by (9) the confirmation message to the client that his request is done. It is up to the UCSs designers to choose the tools, languages, application servers and containers to design their own UCSs tool, but the specifications shown in Figure 5.3 should be considered to meet the architecture requirements. Based on this, any UCSs tool (application/component) created using the new UCSs architecture; should be finally deployed as a composite application in the Service Container (SC). The SC Runtime Environment is provided and supported by most of application servers available these days. SC is integrated with the application server as a pre-configured lifecycle module, which means that whenever the application server's instance starts up, the SC runtime will be available.

There is no user interaction required to configure or start the SC runtime. It is just like any other service of the application server. As shown in Figure 5.3, the Service Engine acts as the bridge between UCSs components and SC. The distributed modules can be packaged in an SC composite application and deployed as one single entity. SC is based on a web services model in most of the available application servers, and provides a pluggable architecture for a container that hosts service producer and consumer components. UCSs administrators (client) and UCSs services connect to the container via binding components or can be hosted inside the container as parts of a service engine.

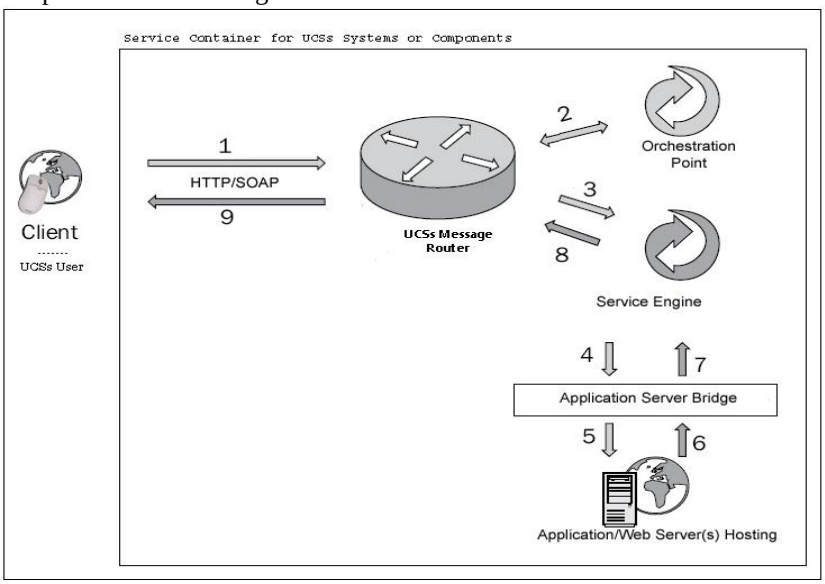


Fig. 4 UCSs composition architecture

 

Furthermore, a Service Container (SC) is a container for structuring business integrated components. It defines an environment for plug-in components that interact using a services model based directly on WSDL. A SC defines a packaging for UCSs composite applications that are composed of UCSs service consumers and providers. Individual service units are deployable to components. Groups of components are gathered together into a service assembly. The service assembly includes meta-data for combining the service units together, as well as binding service units to external services. This provides a simple mechanism for performing composite application assembly using services [20].

## V. A PROTOTYPE BASED ON THE NEW ARCHITECTURE OF UCSS

The specifications of the new architecture constitute a base and a blueprint for developing university systems or components based on the new SOA based architecture described in the previous section. Based on these specifications, this section explores the prototype code of SOA-based UCSs. Java programming language is chosen as the core language to implement this prototype, because it has some advantages over other competitor languages, since it is an open source and platform independent language.

As samples to two components of a UCS, exporting a lecturer time table from the admission and registration system was the first component chosen to validate the new UCSs architecture, while the second component chosen is importing the lecturer time table into the lectures management system. The source code for the ExportLecturerTimeTable.java and ImportLecturerTimeTable.java is shown in table 2.

TABLE II
SOURCE CODE FOR THE EXPORTLECTURERTIMETABLE.JAVA AND IMPORTLECTURERTIMETABLE.JAVA

**ExportLecturerTimeTable class**

```
package ucs.admission;
import java.sql.*;
import java.util.*;
import java.io.*;
public class ExportLecturerTimeTable{
   private String host;
   private String port;
   private String user;
   private String password;
   private String db;
   private String url;
   private Connection conn;
   public ExportLecturerTimeTable (String config) throws
IOException {
      Properties myproperties = new Properties();
      FileInputStream in = new FileInputStream(config);
      myproperties.import(in);
      in.close();
      this.host = myproperties.getProperty("host");
      this.port = myproperties.getProperty("port");
      this.user = myproperties.getProperty("user");
      this.password =
myproperties.getProperty("password");
      this.db = myproperties.getProperty("db");
      this.url = "jdbc:mysql://" + this.host + ":" + this.port
+ "/" + this.db;
   }
   public void jdbcConnect() {
      try {
         Class.forName("com.mysql.jdbc.Driver");
         this.conn = DriverManager.getConnection(this.url,
this.user, this.password);
      } catch (SQLException ex) {
      } catch (ClassNotFoundException ex) {
      } catch (java.lang.Exception ex) {
      }
   }
   public void jdbcConClose() {
      try {
         this.conn.close();
```

**ImportLecturerTimeTable class**

```
package lectures.management;
import java.sql.*;
import java.util.*;
import java.io.*;
public class ImportLecturerTimeTable {
   private String host;
   private String port;
   private String user;
   private String password;
   private String db;
   private String url;
   private Connection conn;
   public ImportLecturerTimeTable (String config)
throws IOException {
      Properties myproperties = new Properties();
      FileInputStream in = new
FileInputStream(config);
      myproperties.import(in);
      in.close();
      this.host = myproperties.getProperty("host");
      this.port = myproperties.getProperty("port");
      this.user = myproperties.getProperty("user");
      this.password =
myproperties.getProperty("password");
      this.db = myproperties.getProperty("db");
      this.url = "jdbc:mysql://" + this.host + ":" +
this.port + "/" + this.db;
   }
   public void jdbcConnect() {
      try {
         Class.forName("com.mysql.jdbc.Driver");
         this.conn =
DriverManager.getConnection(this.url, this.user,
this.password);

      } catch (SQLException ex) {
      } catch (ClassNotFoundException ex) {
      } catch (java.lang.Exception ex) {
      }
   }
```

 

```
        } catch (Exception ex) {
        }
    }
    public void exportTable(String sql) {
        try {
            Statement exportStmt =
this.conn.createStatement();
            exportStmt.executeUpdate(sql);
            exportStmt.close();
        } catch (SQLException ex) {
        }
    }
}
```

```
    public void jdbcConClose() {
        try {
            this.conn.close();
        } catch (Exception ex) {
        }
    }
    public void importTable(String sql) {
        try {
            Statement importStmt =
this.conn.createStatement();
            exportStmt.executeUpdate(sql);
            exportStmt.close();
        } catch (SQLException ex) {
        }
    }
}
```

The WSDL files for the two java components of table 2 are shown in table 3.

TABLE III
WSDL FILES FOR THE TWO JAVA COMPONENTS OF TABLE 2

| WSDL file for the Export Component | WSDL file for the Import Component |
|---|---|
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?><br><!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3.1-hudson-749-SNAPSHOT. --><br><definitions targetNamespace="http://ucs.soa /" name="ExportWSService" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://ucs.soa/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"><br><ns1:Policy wsu:Id="ExportWSPortBinding_ExportOperation_WSAT_Policy" xmlns:ns1="http://www.w3.org/ns/ws-policy"><br><ns1:ExactlyOne><br><ns1:All><br><ns2:ATAlwaysCapability xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/10/wsat"/><br><ns3:ATAssertion ns1:Optional="true" ns4:Optional="true" xmlns:ns4="http://schemas.xmlsoap.org/ws/2002/12/policy" xmlns:ns3="http://schemas.xmlsoap.org/ws/2004/10/wsat"/><br></ns1:All><br></ns1:ExactlyOne><br></ns1:Policy><br><types><br><xsd:schema><br><xsd:import namespace="http://ucs.soa/" schemaLocation="ExportWSService_schema1.xsd"/><br></xsd:schema><br></types><br><message name="ExportOperation"><br><part name="parameters" element="tns:ExportOperation"/><br></message><br><message name="ExportOperationResponse"><br><part name="parameters" element="tns:ExportOperationResponse"/><br></message><br><portType name="ExportWS"><br><operation name="ExportOperation"><br><input message="tns:ExportOperation"/> | <?xml version="1.0" encoding="UTF-8" standalone="yes"?><br><!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3.1-hudson-749-SNAPSHOT. --><br><definitions targetNamespace="http://ucs.soa/" name="ImportWSService" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://ucs.soa/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"><br><ns1:Policy wsu:Id="ImportWSPortBinding_ImportOperation_WSAT_Policy" xmlns:ns1="http://www.w3.org/ns/ws-policy"><br><ns1:ExactlyOne><br><ns1:All><br><ns2:ATAlwaysCapability xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/10/wsat"/><br><ns3:ATAssertion ns1:Optional="true" ns4:Optional="true" xmlns:ns4="http://schemas.xmlsoap.org/ws/2002/12/policy" xmlns:ns3="http://schemas.xmlsoap.org/ws/2004/10/wsat"/><br></ns1:All><br></ns1:ExactlyOne><br></ns1:Policy><br><types><br><xsd:schema><br><xsd:import namespace="http://ucs.soa/" schemaLocation="ImportWSService_schema1.xsd"/><br></xsd:schema><br></types><br><message name="ImportOperation"><br><part name="parameters" element="tns:ImportOperation"/><br></message><br><message name="ImportOperationResponse"><br><part name="parameters" element="tns:ImportOperationResponse"/><br></message><br><portType name="ImportWS"><br><operation name="ImportOperation"><br><input message="tns:ImportOperation"/> |

 

```
        <output
      message="tns:ExportOperationResponse"/>
        </operation>
       </portType>
       <binding name="ExportWSPortBinding"
      type="tns:ExportWS">
        <soap:binding
          transport="http://schemas.xmlsoap.org/soap/http
          " style="document"/>
        <operation name="ExportOperation">
         <ns5:PolicyReference
          URI="#ExportWSPortBinding_ExportOperatio
          n_WSAT_Policy"
          xmlns:ns5="http://www.w3.org/ns/ws-policy"/>
         <soap:operation soapAction=""/>
         <input>
          <soap:body use="literal"/>
         </input>
         <output>
          <soap:body use="literal"/>
         </output>
        </operation>
       </binding>
       <service name="ExportWSService">
        <port name="ExportWSPort"
      binding="tns:ExportWSPortBinding">
         <soap:address
      location="REPLACE_WITH_ACTUAL_URL"/>
        </port>
       </service>
</definitions>
```

```
        <output
      message="tns:ImportOperationResponse"/>
        </operation>
       </portType>
       <binding name="ImportWSPortBinding"
      type="tns:ImportWS">
        <soap:binding
      transport="http://schemas.xmlsoap.org/soap/http"
      style="document"/>
        <operation name="ImportOperation">
         <ns5:PolicyReference
      URI="#ImportWSPortBinding_ImportOperation_W
      SAT_Policy"
      xmlns:ns5="http://www.w3.org/ns/ws-policy"/>
         <soap:operation soapAction=""/>
         <input>
          <soap:body use="literal"/>
         </input>
         <output>
          <soap:body use="literal"/>
         </output>
        </operation>
       </binding>
       <service name="ImportWSService">
        <port name="ImportWSPort"
      binding="tns:ImportWSPortBinding">
         <soap:address
      location="REPLACE_WITH_ACTUAL_URL"/>
        </port>
       </service>
</definitions>
```

Furthermore, the XML schemas for the two components are shown in table 4.

TABLE IV
XML SCHEMAS FOR THE TWO COMPONENTS OF TABLE 2

| XML Schema for the Export Component | XML Schema for the Import Component |
|---|---|
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?><br><xs:schema version="1.0" targetNamespace="http://ucs.soa/" xmlns:tns="http://ucs.soa/" xmlns:xs="http://www.w3.org/2001/XMLSchema"><br><xs:element name="ExportOperation" type="tns:ExportOperation"/><br><xs:element name="ExportOperationResponse" type="tns:ExportOperationResponse"/><br><xs:complexType name="ExportOperation"><br><xs:sequence><br><xs:element name="exportParameter" type="xs:string" minOccurs="0"/><br></xs:sequence><br></xs:complexType><br><xs:complexType name="ExportOperationResponse"><br><xs:sequence><br><xs:element name="return" type="xs:string" minOccurs="0"/><br></xs:sequence><br></xs:complexType><br></xs:schema> | <?xml version="1.0" encoding="UTF-8" standalone="yes"?><br><xs:schema version="1.0" targetNamespace="http://ucs.soa/" xmlns:tns="http://ucs.soa/" xmlns:xs="http://www.w3.org/2001/XMLSchema"><br><xs:element name="ImportOperation" type="tns:ImportOperation"/><br><xs:element name="ImportOperationResponse" type="tns:ImportOperationResponse"/><br><xs:complexType name="ImportOperation"><br><xs:sequence><br><xs:element name="importParameter" type="xs:string" minOccurs="0"/><br></xs:sequence><br></xs:complexType><br><xs:complexType name="ImportOperationResponse"><br><xs:sequence><br><xs:element name="return" type="xs:string" minOccurs="0"/><br></xs:sequence><br></xs:complexType><br></xs:schema> |

As an implementation to the orchestration point that is a mandatory specification of the new UCSs architecture, the BPEL (Business Process Execution Language) was chosen to implement the orchestration point of the prototype. Table 5 shows the implementation code of WSDL and XML files of the BPEL service.

 

TABLE V
IMPLEMENTATION CODE OF WSDL AND XML FILES OF THE BPEL SERVICE

**WSDL File**

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="UCS"
targetNamespace="http://j2ee.netbeans.org/wsdl/UCS_Requestor/UCS_Requestor"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://j2ee.netbeans.org/wsdl/UCS_Requestor/UCS_Requestor"
xmlns:ns="http://xml.netbeans.org/schema/UCS_Requestor"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:tns1="http://j2ee.netbeans.org/wsdl/UCS_Requestor/UCS_Requestor">
   <types>
      <xsd:schema targetNamespace="#TARGET_NAMESPACE">
         <xsd:import namespace="http://xml.netbeans.org/schema/UCS_Requestor"
schemaLocation="UCS_Requestor.xsd"/>
      </xsd:schema>
   </types>
   <message name="UCS_RequestorOperationRequest">
      <part name="requestPart" element="ns:processApplicElement"/>
   </message>
   <message name="UCS_RequestorOperationResponse">
      <part name="responsePart" element="ns:processApplicRespElement"/>
   </message>
   <portType name="UCS_RequestorPortType">
      <operation name="UCS_RequestorOperation">
         <input name="input1" message="tns1:UCS_RequestorOperationRequest"/>
         <output name="output1"
message="tns1:UCS_RequestorOperationResponse"/>
      </operation>
   </portType>
   <plnk:partnerLinkType name="UCS">
         <!-- A partner link type is automatically generated when a new port type is
         added. Partner link types are used by BPEL processes.
         In a BPEL process, a partner link represents the interaction between the
         BPEL process and a partner service. Each partner link is associated with a
         partner link type.
         A partner link type characterizes the conversational relationship between
         two services. The partner link type can have one or two roles.-->
      <plnk:role name="UCS_RequestorPortTypeRole"
portType="tns1:UCS_RequestorPortType"/>
   </plnk:partnerLinkType>
</definitions>
```

**XML File**

```
<?xml version="1.0"
encoding="UTF-8"?>

<xsd:schema
xmlns:xsd="http://www.w3.org/2
001/XMLSchema"

targetNamespace="http://xml.netb
eans.org/schema/UCS_Requestor"

xmlns:tns="http://xml.netbeans.or
g/schema/UCS_Requestor"

elementFormDefault="qualified">
   <xsd:complexType
name="processApplicType">
      <xsd:sequence>
         <xsd:element
name="operationParameter"
nillable="true"
type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType
name="prossApplicRespType">
      <xsd:sequence>
         <xsd:element
name="operationReturn"
nillable="true"
type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element
name="processApplicElement"
type="tns:processApplicType"/>
   <xsd:element
name="processApplicRespEleme
nt"
type="tns:prossApplicRespType"/
>
</xsd:schema>
```

Based on the new UCSs architecture specifications, the two components of the prototype are combined in one deployable composite application. In addition to the theoretical framework specifications, the SOA architecture recommends building loosely coupled applications and treating each one of them as an independent service unit. Well-designed composite applications implement this architectural approach by providing an easy way to build business applications. It also provides integration of existing applications with other existing; as well as new applications. This SOA concept of linking together business processes is the hub of composite applications. Table 6 shows the code of the composite application.

 

TABLE VI
CODE OF THE COMPOSITE APPLICATION

**Code of the composite application**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<casa xmlns="http://java.sun.com/xml/ns/casa" xmlns:ns1="UCS_CompositeApp"
xmlns:ns2="http://j2ee.netbeans.org/wsdl/UCS_Requestor/UCS_Requestor" xmlns:ns3="http://ucs.soa/"
xmlns:ns4="http://enterprise.netbeans.org/bpel/UCS_Requestor/UCS_Requestor"
xmlns:xlink="http://www.w3.org/2000/xlink">
   <endpoints>
      <endpoint endpoint-name="AdministratorPort" interface-name="ns2:UCS_RequestorPortType"
name="endpoint1" service-name="ns1:casaService1"/>
      <endpoint endpoint-name="ExportPort" interface-name="ns3:ExportWS" name="endpoint2" service-
name="ns1:casaService2"/>
      <endpoint display-name="UCSAdministrators_PL" endpoint-
name="UCS_RequestorPortTypeRole_myRole" file-path="UCS_Requestor.bpel" interface-
name="ns2:UCS_RequestorPortType" name="endpoint3" process-name="UCS_Requestor" service-
name="ns4:UCSAdministrators_PL"/>
      <endpoint display-name="Export_PL" endpoint-name="ExportWSRole_partnerRole" file-
path="UCS_Requestor.bpel" interface-name="ns3:ExportWS" name="endpoint4" process-
name="UCS_Requestor" service-name="ns4:Export_PL"/>
      <endpoint display-name="Import_PL" endpoint-name="ImportWSRole_partnerRole" file-
path="UCS_Requestor.bpel" interface-name="ns3:ImportWS" name="endpoint7" process-
name="UCS_Requestor" service-name="ns4:Import_PL"/>
      <endpoint name="endpoint10" endpoint-name="ImportPort" interface-name="ns3:ImportWS" service-
name="ns1:casaService5"/>
   </endpoints>
   <service-units>
      <service-engine-service-unit artifacts-zip="UCS_Requestor.jar" component-name="sun-bpel-engine"
defined="true" description="Represents this Service Unit" internal="true" name="UCS_CompositeApp-
UCS_Requestor" unit-name="UCS_Requestor" unknown="false" x="35" y="99">
         <provides endpoint="endpoint3"/>
         <consumes endpoint="endpoint4"/>
         <consumes endpoint="endpoint5"/>
         <consumes endpoint="endpoint6"/>
         <consumes endpoint="endpoint7"/>
      </service-engine-service-unit>
      <binding-component-service-unit artifacts-zip="sun-http-binding.jar" component-name="sun-http-
binding" description="Represents this Service Unit" name="UCS_CompositeApp-sun-http-binding" unit-
name="sun-http-binding">
         <ports>
            <port bindingType="soap" x="67" y="39">
               <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService1']/port[
@name='AdministratorPort'])" xlink:type="simple"/>
               <consumes endpoint="endpoint1"/>
               <provides endpoint="endpoint1"/>
            </port>
            <port bindingType="soap" x="67" y="118">
               <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService2']/port[
@name='ExportPort'])" xlink:type="simple"/>
               <consumes endpoint="endpoint2"/>
               <provides endpoint="endpoint2"/>
            </port>
            <port x="67" y="197" bindingType="soap">
               <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name=&amp;apos;casaService3&
amp;apos;]/port[@name=&amp;apos;TransformPort&amp;apos;])" xlink:type="simple"/>
               <consumes endpoint="endpoint8"/>
               <provides endpoint="endpoint8"/>
            </port>
            <port x="67" y="276" bindingType="soap">
               <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name=&amp;apos;casaService4&
amp;apos;]/port[@name=&amp;apos;ClassifyPort&amp;apos;])" xlink:type="simple"/>
               <consumes endpoint="endpoint9"/>
```

 

```
                <provides endpoint="endpoint9"/>
              </port>
              <port x="67" y="355" bindingType="soap">
                <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name=&apos;casaService5&
apos;]/port[@name=&apos;ImportPort&apos;])" xlink:type="simple"/>
                <consumes endpoint="endpoint10"/>
                <provides endpoint="endpoint10"/>
              </port>
            </ports>
          </binding-component-service-unit>
        </service-units>
        <connections>
          <connection consumer="endpoint4" provider="endpoint2" state="new"/>
          <connection consumer="endpoint1" provider="endpoint3" state="new"/>
          <connection state="new" consumer="endpoint5" provider="endpoint8"/>
          <connection state="new" consumer="endpoint6" provider="endpoint9"/>
          <connection state="new" consumer="endpoint7" provider="endpoint10"/>
        </connections>
        <porttypes>
          <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/portType[@name='dummyCasaPortTy
pe'])" xlink:type="simple"/>
          <link
xlink:href="../jbiServiceUnits/UCS_Requestor/UCS_Requestor.wsdl#xpointer(/definitions/portType[@name
='UCS_RequestorPortType'])" xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
INF/UCS_Requestor/src/_references/_projects/ClassifyEJB/src/conf/wsdl/ClassifyWSService.wsdl#xpointer
(/definitions/portType[@name='ClassifyWS'])" xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
INF/UCS_Requestor/src/_references/_projects/ImportEJB/src/conf/wsdl/ImportWSService.wsdl#xpointer(/d
efinitions/portType[@name='ImportWS'])" xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
INF/UCS_Requestor/src/_references/_projects/ExportEJB/src/conf/wsdl/ExportWSService.wsdl#xpointer(/d
efinitions/portType[@name='ExportWS'])" xlink:type="simple"/>
          <link
xlink:href="../jbiServiceUnits/UCS_CompositeApp.wsdl#xpointer(/definitions/portType[@name='dummyC
asaPortType'])" xlink:type="simple"/>
        </porttypes>
        <bindings>
          <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/binding[@name='casaBinding1'])"
xlink:type="simple"/>
          <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/binding[@name='casaBinding2'])"
xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
INF/UCS_Requestor/src/_references/_projects/ImportEJB/src/conf/wsdl/ImportWSService.wsdl#xpointer(/d
efinitions/binding[@name='ImportWSPortBinding'])" xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
INF/UCS_Requestor/src/_references/_projects/ExportEJB/src/conf/wsdl/ExportWSService.wsdl#xpointer(/d
efinitions/binding[@name='ExportWSPortBinding'])" xlink:type="simple"/>
          <link
xlink:href="../jbiServiceUnits/UCS_CompositeApp.wsdl#xpointer(/definitions/binding[@name='casaBindin
g1'])" xlink:type="simple"/>
          <link
xlink:href="../jbiServiceUnits/UCS_CompositeApp.wsdl#xpointer(/definitions/binding[@name='casaBindin
g2'])" xlink:type="simple"/>
        </bindings>
        <services>
          <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService1'])"
xlink:type="simple"/>
          <link
xlink:href="../jbiasa/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService2'])"
xlink:type="simple"/>
          <link xlink:href="../jbiServiceUnits/META-
```

 

```
    INF/UCS_Requestor/src/_references/_projects/ClassifyEJB/src/conf/wsdl/ClassifyWSService.wsdl#xpointer
    (/definitions/service[@name='ClassifyWSService'])" xlink:type="simple"/>
        <link xlink:href="../jbiServiceUnits/META-
    INF/UCS_Requestor/src/_references/_projects/ImportEJB/src/conf/wsdl/ImportWSService.wsdl#xpointer(/d
    efinitions/service[@name='ImportWSService'])" xlink:type="simple"/>
        <link xlink:href="../jbiServiceUnits/META-
    INF/UCS_Requestor/src/_references/_projects/TransformEJB/src/conf/wsdl/TransformWSService.wsdl#xpo
    inter(/definitions/service[@name='TransformWSService'])" xlink:type="simple"/>
        <link xlink:href="../jbiServiceUnits/META-
    INF/UCS_Requestor/src/_references/_projects/ExportEJB/src/conf/wsdl/ExportWSService.wsdl#xpointer(/d
    efinitions/service[@name='ExportWSService'])" xlink:type="simple"/>
        <link
    xlink:href="../jbiServiceUnits/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService
    1'])" xlink:type="simple"/>
        <link
    xlink:href="../jbiServiceUnits/UCS_CompositeApp.wsdl#xpointer(/definitions/service[@name='casaService
    2'])" xlink:type="simple"/>
      </services>
      <regions>
        <region name="WSDL Endpoints" width="150"/>
        <region name="JBI Modules" width="904"/>
        <region name="External Modules" width="200"/>
      </regions>
</casa>
```

## VI. Results and Discussion

After developing the SOA-based UCSs prototype; as described in the previous section, this prototype was deployed over two different servers one over Redhat Linux operating systems and the other on Windows 2016 server operating systems. JBoss application server as runtime environments is installed over the server with Redhat Linux operating systems, and open source GlassFish application server is installed over the server with Windows 2016 server operating systems. Each of the two application servers has its own distributed components container which is EJB container.

The ExportLecturerTimeTable component of the prototype is deployed separately on the JBoss application server, while the ImportLecturerTimeTable component of the prototype is deployed separately on the GlassFish application server; the Orchestration point (BPEL Service) is deployed on the other GlassFish application server; and the UCSs administrator (Client) component is deployed separately on the same application server where the Orchestration point deployed on. After that, the execution of the two functionalities is done using browsers of normal client PCs. Therefore, SOA-based UCSs prototype is used here to import and export the lecturer time table. The process is completed successfully while the two components are completely distributed and interoperable.

## VII. Conclusion

This research has explored the common architecture of university computerized systems (UCSs), and has discovered the common gaps and problems available in the architecture of most universities worldwide. In addition it has solved the problems and filled the gaps of the current UCSs by providing the new SOA based UCSs architecture, which was designed and described to ease the understanding of the framework components. In addition, a prototype has validated the new SOA based UCSs architecture. On top of that, this research paper has not only filled the existing distribution and interoperability gaps, the research has also contributed to an increased understanding of how a loosely coupled software framework can be built using service oriented architecture. This is an important issue as more software architectures need to be redefined to eliminate the problems that arise due to the tight coupling feature of the software framework components.

# References


**[1]** F. J. García-Peñalvo, "Digital transformation in the universities: implications of the COVID-19 pandemic," 2021.

**[2]** M. A. Kazemzadeh Raef, M. Masoudinejad, and B. Vasigh, "Comparison of creativity-oriented educational program in the field of architecture of three universities of Tehran, Shahid Beheshti and science and technology," *Creative City Design*, 2022.

**[3]** J. Ma and B. Feng, "Integrated design of graduate education information system of universities in digital campus environment," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

**[4]** G. Rodríguez-Abitia and G. Bribiesca-Correa, "Assessing digital transformation in universities," *Future Internet*, vol. 13, no. 2, p. 52, 2021.


 

**[5]** U. U. Malaysia. (2022, 23-02-2022). *The official University website*. Available: https://uum.edu.my/
**[6]** L. U. o. C. Technology. (2022, 15-02-2022). *The University official website*. Available: https://www.limkokwing.net/
**[7]** U. o. Palestine. (2022, 02-02-2022). *The University official website*. Available: https://up.edu.ps/ar/
**[8]** M. Ke, "Construction of Enterprise Operation Simulation System Platform based on Service Oriented Architecture," in *Proceedings of the 3rd Asia-Pacific Conference on Image Processing, Electronics and Computers*, 2022, pp. 955-959.
**[9]** K. Zaafouri, M. Chaabane, and I. B. Rodriguez, "Systematic Literature Review on Service Oriented Architecture Modeling," in *International Conference on Computational Science and Its Applications*, 2021, pp. 201-210: Springer.
**[10]** E. Hustad and D. H. Olsen, "Creating a sustainable digital infrastructure: The role of service-oriented architecture," *Procedia Computer Science*, vol. 181, pp. 597-604, 2021.
**[11]** H. Rojas, K. A. Arias, and R. Renteria, "Service-oriented architecture design for small and medium enterprises with infrastructure and cost optimization," *Procedia Computer Science*, vol. 179, pp. 488-497, 2021.
**[12]** V. Issarny, C. Kloukinas, A. Zarras, and M. Architectures, "Management Group's Common Object Request Broker (CORBA)," *Microsoft's Distributed Component Object Model*, 2008.
**[13]** J. Maassen, R. Nieuwpoort, R. Veldema, H. E. Bal, and A. Plaat, *Java Remote Method Invocation provides an unusually flexiblilty*. Indiana, USA: Wiley Publishing, Inc., 2008.
**[14]** M. Tsenov, "Example of communication between distributed network systems using web services," *Proceedings of the 2007 international conference on Computer systems and technologies*, 2007.
**[15]** G. R. Voth, C. Kindel, and J. Fujioka, "Distributed application development for three-tier architectures: Microsoft on Windows DNA," *IEEE Internet Computing*, vol. 2, no. 2, pp. 41-45, 1998.
**[16]** B. G. Sullins and M. B. Whipple, *EJB Cookbook*, 1st ed. Philadelphia, USA: Manning, 2005.
**[17]** F. Perin, "Enabling the Evolution of J2EE Applications through Reverse Engineering and Quality Assurance," *Proceedings of the 2009 16th Working Conference on Reverse Engineering*, pp. 291-294, 2009.
**[18]** P. Louridas, "SOAP and Web Services," *IEEE Software*, 2006.
**[19]** H. Szczepaniuk, "Distributed Internet of Things applications in a trust-based Service-Oriented Architecture," in *Trust, Digital Business and Technology*: Routledge, 2022, pp. 107-118.
**[20]** D. Salter and F. Jennings, *Building SOA-Based Composite Applications Using NetBeans IDE 6*, 1st ed. USA: PACKT Publishing, 2008.

 