**Chapter Five**

**EVALUATION OF PROPOSED ARCHITECTURE**

**1. Introduction**

This chapter answers the many questions arose about SOA efficiency and capabilities regarding overcoming UMIS and LMS limitations and presents the results of evaluating the proposed UMIS and LMS SOA based components. From information system point of view, points like performance, integration and interoperability, compliance, security, maintainability, analyzability, decomposability and modularity, testability, portability via replaceability and scalability, simplicity, modifiability, and reusability are addressed. A Comparative performance analysis study is available to test SOA based systems user-perceived performance against non-SOA based systems. Pedagogically, SOA adoption needs to enhance the learning process activities and provide capabilities that was hard to present before.

**2. Performance**

One of the main reasons to focus on styles for network-based applications is because component interactions can be the dominant factor in determining user-perceived performance and network efficiency. Since the architectural style influences the nature of those interactions, selection of an appropriate architectural style can make the difference between success and failure in the deployment of a service-based application. Performance refers to both Network and User-perceived performance.

**2.1 Network Performance**

Service Oriented Architecture based application relies heavily on messaging. Debates that relying on messaging causes delays more than non-messaging applications. Besides, it is clear now that SOA based applications need to add extra headers to manage requests and responses in standard format. Network performance of proposed SOA based architecture was evaluated against non-SOA based architecture.

**Types of Delay in Packet-Switched Networks are:**

* **Processing Delay**: The time required to examine the packet’s header and determine where to direct the packet is part of the processing delay. The processing delay can also include other factors, such as the time needed to check for bit-level errors in that packet that occurred in transmitting the packet’s bits from the upstream node to router A. Processing delays in high-speed routers are typically on the order of microseconds or less.
* **Queuing Delay:** At the queue, the packet experiences a queuing delay as it waits to be transmitted on the link. The length of the queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission across the link. If the queue is empty and no other packet is currently being transmitted, then packet’s queuing will be zero.  On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long. The number of packets that an arriving packet might expect to find is a function of the intensity and nature of the traffic arriving at the queue. Queuing delays can be on the order of microseconds to milliseconds in practice.
* **Transmission Delay:** Assuming that packets are transmitted in a first-come-first-served manner, as is common in packet-switched networks; packet can be transmitted only after all the packets that have arrived before it have been transmitted. Denote the length of the packet by *L* bits, and denote the transmission rate of the link from router A to router B by *R* bits/sec. The rate *R* is determined by the transmission rate of the link from router A to router B. The transmission delay (also called the store-and-forward delay) is *L/R.* This is the amount of time that is required to push (that is, transmit) all of the packets bits into the link. Transmission delays are typically on the order of microseconds to milliseconds in practice.
* **Propagation Delay:** Once a bit is pushed onto the link, it needs to propagate to router B. the time required to propagate from the beginning of the link to router B is the propagation delay. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link. The propagation delay is the distance between two routers divided by the propagation speed.

Figure 5.1 shows one of the services extra header added by the service as the Request, while figure 5.2 shows the response format. The request and response are related to Insert Author service; that is one of the service designed and implemented in LIS.

Figure 5.1 includes text within dashed boxes that represent static header added every time when the service is invoked. Text outside dashed boxes presents header added once for every record. That means, based on what is depicted in figure 5.1, when there will be one author to add, static header will be repeated once for the request (that is the header inside dashed box) while header outside the dashed box will be repeated as many times as the number of authors will be inserted. Values (int, string, string, int) are replaced with real life author values.

POST /desktop\_library/auther\_desktop\_library/Service.asmx HTTP/1.1

Host: localhost

Content-Type: text/xml; charset=utf-8

Content-Length: length

SOAPAction: "http://tempuri.org/auther\_Insert"

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>

    <author\_Insert xmlns="http://tempuri.org/">

      <author\_id>int</author\_id>

      <author\_name>string</author\_name>

      <nationality>string</nationality>

      <no\_authrd\_book>int</no\_authrd\_book>

    </author\_Insert>

  </soap:Body>

</soap:Envelope>

**Figure 5.1: Insert Author SOAP Request**

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Body>

    <author\_InsertResponse xmlns="http://tempuri.org/">

      <author\_InsertResult>bool</author\_InsertResult>

    </author\_InsertResponse>

  </soap:Body>

</soap:Envelope>

**Figure 5.2: Insert Author SOAP Response**

This comparative network analysis note will rely on the delay calculation formula, where:

**Total Delay = Transmission delay + Propagation delay + Processing delay + Queuing delay**

But it is well known that processing and queuing delay are less than micro seconds, so they are ignored, so the delay formula will be:

**Total Delay = Transmission Delay + Propagation Delay**

Transmission delay, where:

Dtrans = (M + N – 1) L / R

Where:

* M = no. of communication links
* N = no of packets
* L = packet size
* R = Transmission Rate

Transmission delay is affected by file size (F). In the previous formula F = N\*L.

By analyzing data in figure 5.1, it is noticeable that there are three data categories:

* **Static Header:** This header occurs once for each service invocation no matter how many records in the request. Characters for this header = 463 characters
* **XML Tags:** Those tags are the overload of requests and responses. Those tags are named by developer, so they are not static every time, but in the presented example there are 179 characters
* **Actual Data:** Those are the record details to be inserted after invoking the Insert Web service.

So, Total extra characters to insert an author = Static Header + XML Tags (642 characters; more than half a packet size). So, file size in SOA will be:

FSOA = F + SH + RH \* R

Where:

* F = total data size without headers
* SH = Static Header
* RH = XML tags required to represent a single record
* R = no of records

This formula depicts that added extra headers differs according to the no. of records to be handled, and differs from an application to another (because the header used to represent an author might be different from the one used to represent a book) so network performance differs from an application to another. It is the system architect responsibility to decrease the transferred data over the network to the maximum extent (so decrease network delay) because it is clear now that the added headers by SOA is not the headers that can be neglected easily.

* 1. **User Perceived Performance**

User-perceived performance differs from network performance in that the performance of an action is measured in terms of its impact on the user in front of an application rather than the rate at which the network moves information. The primary measures for user-perceived performance is completion time. Completion is the amount of time taken to complete an application action. SOA based Systems is evaluated regarding User-perceived performance against other two well known architectures. Three different Library Management System architectures were implemented and user-perceived performance was measured against same sample data and against same queries.

**Evaluated Architectures**

Three Library Management System (LIS) architectures were evaluated. The first one is the Services based LIS architecture presented in Chapter 3. The other two Library Management Systems are: Parametrized Query, and Stored Procedure based architectures. Figure 5.3 presents the Parametrized Query based architecture where SQL statements exist within the web pages and accesses database directly. Figure 5.4 highlights the separation between data layer and application layer by the presence of Stored Procedures as a middle layer in-between portal and database. Portal consumes stored procedures to access databases. Figure 5.5 shows the services based architecture with the services layer available in between the portal and database layer to present a standard based interface layer that consumes stored procedures and available for portals. Table 5.1 presents the Advantages and disadvantages of Parameterized Query adaption in software systems. Table 5.2 shows the advantages and disadvantages of Stored Procedure based software architectures. Services based architectures advantages were intensively discussed in chapter 2.



**Figure 5.3: Parameterized Query Based LIS Architecture**



**Figure 5.4: Stored Procedure Based LIS Architecture**



**Figure 5.5: Services Based LIS Architecture**

**Table 5.1: Advantages and Disadvantages of Parameterized Quereis**

|  |  |
| --- | --- |
| Parameterized Queries | |
| Advantages | **Dis-Advantages** |
| Code reusability for different tables, objects, and databases | Hard to debug, and error management becomes more unreliable |
| Use variable names in statements that require constants | Temporary tables from the main statement cannot be used, unless they are global |
| Return ROWSETs with a variable number of columns and / or variable column names | Algorithm time adds up to the time of dynamic SQL execution |
| Sorting by any column from a table | Difficulty of maintenance is difficult schema is hard coding in the dynamic code |
|  | Security can be comprised with SQL injection |

**Table 5.2: Advantages and Disadvantages of Stored Procedures**

|  |  |
| --- | --- |
| Stored Procedures | |
| Advantages | **Dis-Advantages** |
| Improve security of database server | Slower than equivalent application code |
| Offer a mechanism to abstract data access routines which can improve code maintainability | Lead to application logic fragmentation between database and application tier |
| Reduce network traffic | Difficult to debug |
| Implement common routines and make them accessible from multiple applications | Most object-relational mapping systems cannot seamlessly exploit stored procedures |
| Database centric logic can be isolated |  |
| Improve application portability by moving data logic into stored procedures |  |

**Analysis Tools**

More than one analysis tool has been used while evaluating and analyzing proposed architectures performance. Analysis tools include, not only, Firebug; an add-on Firefox. Firebug integrates with Firefox to put a wealth of web development tools while the system developer browses web based systems. Editing, debugging, and monitoring browsed web pages are applicable. Main features of Firebug include the capability to: Inspect and edit HTML, Visualize CSS metrics, Monitor Network activity, Debug and profile JavaScript, measure performance and find bottlenecks fast, Quickly find errors, Explore the DOM, Execute JavaScript on the fly, and Log for JavaScript [20,21].

The most interesting feature of Firebug is the capability to watch the timeline of the web page unfolds. Firebug shows a bar for each file that displays when the file started and stopped loading relative to all the other files. Another interesting feature is Firebug’s capability to Examine HTTP Headers that contain important information like the file mime type, web server type, caching directives, and cookie. HTTP headers have been used heavily in Services-based Architecture.

Firebug was used heavily in the developing process based on presented functionalities, but in the analysis process it was used mainly to measure the timeline of page unfolds so measure the user perceived performance of the three LIS actitectures.

**Performance Measures**

Arithmetic Mean (Average) is the value obtained by dividing the set of quantities by the number of quantities in the set, so the Arithmetic Mean for each presented architecture regarding each presented process = Total time consumed to perform operation / Total no. of records. Another statistical measure that is used to evaluate the performance of presented architecture is Mode, which is the Most Repeated Value for each operation. Figure 5.6 illustrates statistics of the Insert process for the three implemented Library Management System architectures. Table 5.3 is a summary of the arithmetic mean and mode for each architecture. Services based architecture presented the highest arithmetic mean and mode values with no much performance enhancement to be mentioned. Stored Procedure based architecture was the best for the insert operation.

**Figure 5.6: Insert Performance Measures of the three LIS Architectures**

**Table 5.3: Insert Operation Measurements Summary – Measures are in Milli-Seconds per Record**

|  |  |  |
| --- | --- | --- |
| Architecture | Arithmetic Mean | Mode |
| Service Based | 137.1 | 100 |
| Stored Procedure Based | 69.2 | 50 |
| Parameterized Query Based | 76.2 | 70 |

Figure 5.7 presents the statistics of the Update operation performed by the three implemented Library Management System architectures followed by table 5.4 that displays summary of arithmetic mean and average of the same operation. Services based architecture is the lowest in performance compared to stored procedure and parameterized query based architectures. Stored Procedures based architecture, is the best performance of the Update operation.

**Figure 5.7: Update Performance Measures of the three LIS Architectures**

**Table 5.4: Update Operation Measurements Summary – Measures are in Milli-Seconds per Record**

|  |  |  |
| --- | --- | --- |
| Architecture | Arithmetic Mean | Mode |
| Service Based | 141.2 | 120 |
| Stored Procedure Based | 103 | 80 |
| Parameterized Query Based | 109.3 | 70 |

Figure 5.8 presents statistics of the Select By ID operation of the three implemented Library Management System architectures, followed by table 5.5 that summarizes the arithmetic mean and mode of the three presented performance measures. Services based architecture is the highest in ranges. While arithmetic mean and mode depicts that parameterized query based architecture performance is better than the stored procedure based one, it is noticed that parameterized query based architecture was highly affected by the amount of data retrieved, and its performance was not within small ranges; not like stored procedure one.

**Figure 5.8: Select By ID Performance Measures of the three LIS Architectures**

**Table 5.5: Select By ID Operation Measurements Summary – Measures are in Milli-Seconds per Record**

|  |  |  |
| --- | --- | --- |
| Architecture | Arithmetic Mean | Mode |
| Service Based | 143 | 110 |
| Stored Procedure Based | 113 | 100 |
| Parameterized Query Based | 112.1 | 90 |

Figure 5.9 presents the total amount of time required by each of the three implemented Library Management System architecture to retrieve all data stored in the database, with no filter applied. Stored Procedure based architecture achieved the best time, services based architecture required time to retrieve the stored records exceeded the double time consumed by stored procedure based system, and parameterized query based system performance lies in between.

**Figure 5.9: Display All Performance Analysis of the three LIS Architectures**

**Comments on Results**

The total time required by LIS to satisfy user request for any operation since user initiates request using the sample data is:

Total Elapsed Time = Network Time + Processing Time + Display Time

1. Network Time; include:

* Time to Send Request from User to Web server
* Time to send DB request from Web server to DB server
* Time to send reply from DB Server to Web server
* Time to send Response from Web server to User

Because requests were initiated from the same PC each time, and the database exists on the same application server, network delay is almost equal in all requests; ignoring states of temporarily congestion, so network consumed time is not the main reason for performance differences between the three architectures.

1. Processing Time; include:

* Web server processing of the sent request pages (include: Parameters processing)
* Web server preparing the request (incase Services are used)
* SQL Server processing of the request (insert, update, and retrieve operations)
* Web server processing of the response

Though generally, processing time is neglected in almost all application evaluation strategies due to the huge revolution in hardware computing and the presence of new highly performance technologies, but performance in the services based architecture was affected. The difference in delays between the three architectures is the result of performance delays. Web services consumption, extra HTTP headers processing, preparing requests and waiting for responses result in the highlighted performance measures differences.

1. Display Time

* Time to display web page that holds response data on the user screen.

The returned data size is the same for each request applied to the three different architectures. The Utilized analysis tool allowed the unfolding of the response page that holds the processed data from the rest of the portal interface regarding display time, so the Display time is not affected by the architecture implementation at all.

From the performance analysis presented and after evaluation of the three architectures, it is clear that the time consumed to perform the same operations using the services based architecture exceeds the time consumed to perform the same operation using either the stored procedure architecture, or the parameterized query one. But it should be clear that while other architectures might perform faster than SOA based applications, those architectures can not do what SOA does.

1. **Functionality**

Functionality evaluation focuses on the existence of a set of functions and their specified properties. Functions are those that satisfy stated or implied needs.

**3.1 Integration and Interoperability**

A slight difference between integration and interoperability exists, where integration means the capability of an application to perform its tasks without the need from other application, but still the fact that results of performing those tasks needs to be shared with other applications. On the other hand, interoperability refers to the application need to communicate with other applications to achieve the task.

It is really worth the effort to see that systems can share their effects within a single operation by integrating them on service level, not just on data or application level. Assessment Management System did not have to access Student Affairs Information System database tables to retrieve and update student table data; instead, it just invoked the Update\_Student service exposed by it.

Assessment Management System is a real example on the interoperability extent that SOA based applications can achieve. The Take Assessment Process needs interoperability between Assessment Management System and external systems. Without this interoperability, Mobile assessment would not have taken place at all. SOA utilization in the system gave the system capability to expose standard interfaces that act like sockets to be plugged in to connect systems. We did not even needed system architects from the mobile company to manage the application, we consumed the standard interface they exposed and Assessment Management System were able to provide the new Mobile assessment process.

**3.2 Compliance**

Compliance refers to the software adherence to application related standards of conventions or regulations in laws and similar prescriptions. Proposed SOA based system implemented the abstraction and separation of layer concerns. Orchestration and application services layer were separated to achieve the compliance goal. It was clear in the Assessment Management System when the learner had to take an assessment within 24 hours; this simple data was stored explicitly in the database and managed by the Manage Assessment service. When the system needs to change the period of which the student can take the assessment, changes are not made to the application, neither at application services layer, nor at orchestras.

**3.3 Security**

Security refers to software ability to prevent unauthorized access, whether accidental or deliberate, to programs or data. Classical SOA; by default; utilizes certain security hints that enhance the overall system security, they include:

* The first step for security is to hide the existence of the service from the attackers, and this can happen easily via SOA. Middleware can be placed in the middle between consumer and the service, this middleware guides the request to the service while protecting it from attacks.
* Web services make use of the enhancements performed by Web services manufacturers (Microsoft and IBM for example). Web services 2.0 and Web Services Enhancements (WSE) are examples of by default acquired advantages of implementing Web services overtime.
* Web services act as an isolating layer of database; there is no direct access to the database available, and operations are stated within the service. The service requestor does not even need to have an account on the database.

**4. Maintainability**

Maintainability refers to the ease and speed with which systems can be understood and modified.  
 **4.1 Analyzability, Decomposability, and Modularity**

Analyzability is the effort needed for diagnosis of deficiencies or causes of failures or for identification of parts to be modified. Decomposability isthe process of breaking down a system into its smaller components. These components may themselves be systems (subsystems) and can be broken down into their components as well. This decomposition results in smaller and less complex pieces that are easier to understand than larger, complicated pieces. Decomposing a system allows analyst to focus on one part of the system, making it easier to think of how to modify that one part independent of the entire system Modularity is a direct result of decomposability. It refers to dividing the system into chunks or modules of a relatively uniform size. Modules can represent a system simply, making it easier to understand and easier to redesign and rebuild. Composing systems into stand alone services makes it more analyzable, because it gets decomposable and modular. SOA based applications are easier to determine areas of failure.

**4.2 Testability**

Software testing is the process used to measure the quality of developed software. Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors.

**Levels of Testing**

* **Unit Testing:** tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an Object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

The minimal software component or module is: Service. Each service can be easily tested by itself to assure it performs the required functionality. WSDL provides a standard interface that enables service test even without the need to write a testing application. Applying SOA saves time of testing so, there can be completely new systems build without the need to apply unit testing, because they utilize previously tested services.

* **Functional Testing:** tests at any level (class, module, interface, or system) for proper functionality as defined in the specification.

Functional test needs to be applied to units and system level. By applying functionality test to services via easy by default designed services based unit test, it is clear that the available unit test facilitated the functional testing.

* **Integration Testing:** exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

Because proposed SOA based system utilizes Web services as the main SOA enabler, and because Web services utilizes standard protocols set, integration shortages and limitations were overcome. Integration requirements are not hard to design, implement, and tested incase it is based on Web services based SOA. Integration has turned to be the art of connecting the proper communication points to each other within systems.

**5. Portability**

Portability is the ability of software to be transferred from one environment to another.

**5.1 Replaceability**

It is the opportunity and effort using the software in the place of specified other software in the environment of that software. One of the main advantages of SOA is that it enables replaceability of services utilized from outside incase that the new utilized services maintain the same service interface. It will take just changing the address URI to the new address to utilize the new service.

**6. Scalability**

Scalability isthe ease with which a system or component can be modified to fit the problem area. Anarchic Scalability refers to the need for architectural elements to continue operating when they are subjected to an anticipated load, or when given malformed or maliciously constructed data, since they may be communicating with elements outside their organizational control. The architecture must be amenable to mechanisms that enhance scalability. Scalability can be achieved via hardware, or software, or both.

**6.1 Hardware Scalability**

**Modify Hardware Requirements**

Highly rated requested services can be separated on stand alone servers specialized for those services in order to scale the overall system. If this option is not applicable, the servers that hold the highly rated requested services can get hardware upgrades faster than the rest of the system, so the overall system performance can be enhanced.

**Load Balancing**

Scalability can be achieved by Load Balancing Servers. Load Balancing Servers are the servers responsible for balancing loads between multiple servers hold the same required services. Figure 5.10 depicts the classical Web service consumption scenario that is the one without a load balancing server as depicted in figure 5.11. In figure 5.11, the load balancing server is responsible for balancing the requests between the two application servers that hold the same service.



**Figure 5.10: Classical Web service Consumption Scenario**



**Figure 5.11: Load Balanced Web service Consumption Scenario**

**6.2 Software Scalability**

* System modularity enables it to be enhanced service by service so the system lifecycle is enhanced on the software scalability level. Services that highly rated requested can be enhanced, and optimized, so the overall system will be enhanced.
* Web services based SOA is immune to malicious software attacks because exchanged messages between requestors and services providers do not hold code to be executed, it only holds XML based data. Messages are not instructive, they are directive.

**7. Simplicity**

The primary means by which architectural styles induce simplicity is by applying the principle of separation of concerns to the allocation of functionality within components.

Separation of concerns into layers is what enables SOA to be simple. Layers enable the system to be understandable and less complex because by checking in which layer the service exists gives a general overview of the type of this service functionality.

**8. Modifiability**

Modifiability is about the ease with which a change can be made to application architecture. A particular concern of network-based systems is dynamic modifiability, where the modification is made to a deployed application without stopping and restarting the entire system. Even if it were possible to build software system that perfectly matches the requirements of its users, those requirements will change over time just as society changes over time.

**8.1 Extensibility**

Extensibility was tested in the proposed architecture by adding functionalities to the system that required adding new services that did not exist in the first place. Because services are loosely coupled, they can be added and consumed without affecting any other services. New added services can utilize existing services, consume databases and stored procedures, wrap legacy systems, and any functionality required without affecting the existing system.

**8.2 Reusability**

Reusability is a property of application architecture if its components, connectors, or data elements can be reused, without modification, in other applications. It the practice that a segment of source code can be used again to add new functionalities with slight or no modification. IEEE 90 defines it as the degree to which a software module or other work product can be used in more than one computing program or software system. Reusable module and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modification when a change in implementation is required.

Reusability is achieved in the proposed architecture on two levels: Internal and External. Internal reusability refers to the application capability to use the same implemented functionality more than one time without modification. This happened for example with the Update functionality, where it consumed Delete and Insert functions. Besides, functions were not written every time from the start, on the other hand, they were recalled from the implemented service. External Reusability refers to the external systems that consumed the exposed internal services to achieve functionalities.

**9. Pedagogical Evaluation**

Incase architectural and information systems aspects of University Management Systems and Learning Management Systems needed the SOA based system, pedagogical aspects are affected by the proposed SOA based architecture, at least indirectly. Learning Management Systems might think that there is no more functionality to be provided, but SOA proved there is still.

* E-Learning solutions was not affected; from the pedagogical point of view, by adapting SOA. There was no pedagogical requirement that was satisfied by non-SOA Learning Management Systems that was not available when SOA was adopted.
* SOA adoption facilitated integration of software agents within new proposed systems. Software agents have played; and still, major roles in e-Learning. Integrating software agents with Web services was presented successfully in proposed Course Management System and Assessment Management System as an example of the success of integration.
* Mobile assessment refers to the capability of conducting assessments via mobile devices. Mobile assessment relies on external services that are not part of the LMS. Integrating different external systems and services to be virtually part of the educational institution LMS is one of integration challenges. Mobile Learning (M-Learning) is an approach to electronic learning (E-learning) that simply utilizes mobile devices, yet it can also be viewed as a quiet different learning experience (Hulme and John 2005). It is possible to force series of interactive SMS exchanges between learner and LMS to achieve completion of a task or goal. Learner will take part, and complete the task. M-learning has been used as a pre and/or post activity to other types of learning. M-learning has been widely considered and implemented. Assessment for learning can be thought as one of the post learning activity that can be achieved via mobile phones. Mobile Learning was successfully implemented and the main enabler was SOA adoption.
* Unlocking Course Repositories “Automating the Discovery, Downloading, and Paying of Shared Courses”: One of the critical limitations of a newly established educational institution is the lack of available well prepared courses. It is more applicable to use widely available courses that might be higher in quality than preparing new courses. Current course management systems do not exploit courses shareability. To address this shortage, a Course Management System (CMS) is proposed to highlight automated discovering and importing of courses maintained and managed by external CMSs. It is generally a hard job for instructor to prepare electronic course contents. Today’s commercial CMSs do not address the capability of automated search and import of external courses, especially if courses are not free. Courses are about quality, not quantity. Departments want fewer modules that incorporate simulations and interactivity. Educational institutions can increase Return-On-Investment (ROI) by selling courses.
* Digital Library contents are available to all LMS components to utilize, search within, and enrich the learning activity with valuable contents without the need to adopt new systems. SOA facilitated the integration between LMS components and Digital Library solution.

1. **Summary**

This chapter presented an evaluation framework that can be used to evaluate LMS. Proposed evaluation framework attempted to evaluate LMS from more than one perspective, so a general overview of the proposed LMS can be achieved. Proposed evaluation framework can be used as a whole, or by topic.

SOA adoption within e-Learning in the form of University Management System and Learning Management System presented information systems’ advantages as well as pedagogical ones. It is clear that there is still more to be discovered and more advantages will become available upon adopting SOA in e-Learning.

Pedagogically, SOA has helped e-Learning achieve more than one goal. One of the critical limitations of a newly established educational institution is the lack of available well prepared courses. It is more applicable to use widely available courses that might be higher in quality than preparing new courses. Current LMS do not exploit courses shareability. Proposed SOA based LMS addressed this shortage, automated discovering and importing of courses maintained and managed by external LMSs. Proposed LMS facilitates integration between different LMSs in order to share resources of educational institutions.

SOA facilitated integration between software agents that play an important role in educational institutions and Web services; that is the core of proposed SOA LMS. Also, integrating legacy systems and newly added systems is facilitated by SOA.

Also, Mobile Learning (M-Learning); which is an approach to E-Learning that utilizes mobile devices is enabled by proposed LMS. M-Learning can be adapted via SOA based LMS. Mobile assessment is one of the M-Learning activities facilitated by proposed SOA based architecture. Mobile assessment relies on external services that are not part of the LMS. Integrating different external systems and services to be virtually part of the educational institution LMS is one of integration challenges. The capability to integrate the different digital library contents and make it available to different LMS components is a clear example of the SOA capabilities to integrate different and standalone system components and make them available to each other.