Journal of Theoretical and Applied Information Technology
15th April 2020. Vol.98. No 07

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SOFTWARE ARCHITECTURE USING A DECOUPLING, LAYERED AND WEB RESTFUL SERVICES APPROACH

[1]SALAS-PARTIDA RUBEN, [2]RODRIGUEZ-AVILA EDUARDO, [3]RAMIREZ-ARELLANO ALDO, [4]ACOSTA-GONZAGA ELIZABETH

Instituto Politécnico Nacional

Av. Té 950, Granjas México, 08500, Ciudad de México, México

E-mail: [1]partida0001@gmail.com, [2]errodriguez@ipn.mx, [3]aramirezar@ipn.mx, [4]eacostag@ipn.mx

ABSTRACT

The development of information systems has forced the software industry to create architectures, techniques, and tools for building applications more efficiently. This work shows the design of information systems using code decoupling, layered development, and RESTful services. Two information systems were developed, one using coupled methods and the other using decoupled methods. To test both methods, two experiments were performed for each system, one using a reduced data model for accessing four database tables, and the other experiment allowing only the execution of basic mathematical operations. The results show that a decoupled architecture offers advantages in response time, network traffic, and user concurrency, compared to those built without these techniques. This work is limited to the Java programming language and the specification Java Enterprise Edition.

Keywords: *Software Architecture, Coupling Software, Decoupling Software, Web Restful Services, Software Metrics.*

1. INTRODUCTION

Designing a software architecture for information systems is complex work. A system can be built on a single layer (standalone), two layers (on-line, client-database) or three layers (web model, client-server-database), still its design and development will require an analysis of the functional context and the requirements to be satisfied [1].

The design starts from defining the layers that the information system will use, including the degree of software coupling with purpose of reuse and maintenance. Programming the system once and using it many times is the goal of modern software engineering practices, which also will reduce the cost of developing complex information systems and other future developments from these [2].

The objective of this work is to show that information systems development using a decoupling code, layered architecture, and RESTful services approach offers greater benefits compared to software development approaches without these.

To verify the objective, two experiments were designed to compare two software systems developed in Java; both had the same data source and were developed with the same hardware and software platform. One application was developed using a coupling code approach, and the other using a decoupling code approach.

The results show that the development of software applications using decoupling software design offers more benefits in performance and maintainability, compare to software development that does not use these approaches.

The article has been organized as follows. Firstly, the related work is presented. Next, the theoretical framework defines key concepts on which the work is based. Finally, the experimental design, results, and conclusions are presented.

2. RELATED WORK

In the earliest days of computing, when programmers directly loaded instructions on the computer memory, the appearance of certain patterns and structures was noticed. The presence of these patterns and structures promoted the usage of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

high-level assemblers and languages. Since then, the idea of developing an efficient structure or structures that would facilitate the development, maintenance or extension of the programs was already shared between analysts and programmers. The idea was transformed into theories and so the research on software architecture, which concept was developed in the late 1960s, with Edsger Dijkstra's research [3] and the early 1970s, with David Parnas's works [4]. Since then, the need to adopt design principles or formal methods for software development has grown and these activities have increased their complexity. In fact, in every information system there is an intentional design of software architecture, or not [5].

The literature review of previous contributions suggests three areas of work related to software architecture and information systems. The first is related to applications [6–12]. The second area refers to the state of art and discussions [13–15]. The third area deals with proposals for model development [16–21].

The design of modern software architecture does not only include aspects of software costs, maintenance, and reuse, it must be technologically independent [22]. Recent works have put architectural considerations on coupling as an important metric of the software architecture quality [23, 24] and modularization [25]. Other researchers have focused on studies about coupling metrics [26, 27]. In particular, Gui and Scott [26, 27] proposed a set of coupling and cohesion metrics to assess the reusability of Java components. Bidve and Sarasu [28] investigated a correlation between the values of coupling metrics and the number of classes in the multimedia Java code.

Researchers such as Zhe and Kerong [29] proposed a software partition decoupling method based on an interactive genetic algorithm. Bidve and Sarasu [28] presented a tool that analyses the Java source code to measure coupling among various Java software modules.

Recent research has been focused on decoupling code approaches [30], the use of RESTful services [31], the decoupling degree as a measure of architecture [23], and the emphasis on decoupled architectures [32].

3. THEORETICAL FRAMEWORK

The following subsections define some concepts related to information systems software development. A common issue in software development is the adoption of technological trends or fashionable designs [33]. However, an efficient approach allows not only to understand why certain decisions are made but also guarantees that these decisions will offer greater stability to the system when changes occur.

3.1. Software architecture

Fielding [33] proposes a software architecture considering three key elements to define a set of architectural properties. The first key element is the software component, which is seen as instruction units and states that work can be isolated without depending on the data from other components, but also providing data transference. The second element is the connector, which is an abstract mechanism that coordinate communication between components. Finally, the third element is the data flowing between components via the connectors.

Thus, from this perspective, the software architecture for information systems is the result of the abstraction at runtime of its elements at some point in its operation [33]. The purpose for implementing an architectural design is not only that information systems operate with the best performance and distribute the processes appropriately in the layers, but also that it facilitates its development, integration, and maintenance.

3.2. Abstraction at Runtime

The basis of the abstraction at runtime is to hide details of the operation of several software components, understanding that each component performs a specific function and exposes its interface with other components [33, 34].

3.3. Software Architectural Patterns

A software architecture pattern is an already proven solution that works to solve a given problem [35]. Architectural patterns and frameworks are proven techniques and tools that allow the agile development of an application [1].

3.4. Layered Or Multilayered Development

The information systems development is divided into three main layers: the data access layer, the business layer, and the presentation layer [36]. Each of these layers currently uses different frameworks, depending on the platform used to execute the application [1].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.5. Systems Integration

Systems integration refers to interconnecting two or more information systems, that, while they are independent, still require access to data or processes from one system by another. These systems may be running within the Intranet, or they may be running external to it. The communication between two systems can be synchronous or asynchronous. In the following, some systems integration technologies are described.

The Java Message Service (JMS) serves to communicate two or more systems asynchronously. Among the systems, two types of destination occur: (1) queues that send messages from a producer to a consumer, queuing them with a first in, first out (FIFO) technique; (2) topic, which is the subscriber producer pattern and in which the producer sends a message to several subscribers.

A web service is a communication mechanism between systems, processes or applications, that uses the HTTP protocol to transfer information. This mechanism uses a Simple Object Access Protocol (SOAP), which defines how two objects belonging to different processes can communicate with one another by means of message exchange in Extensible Markup Language (XML) format.

The REpresentational State Transfer (REST) defines a software architecture design that relies on the transfer of messages without saving any state or information between the requests of messages.

The RESTful Web is based on the REST architecture. It is a stateless model, in which the client makes a request and obtains a response from the server. It accesses to Uniform Resource Identifier (URI), and uses the HTTP protocol and its operations POST, PUT, DELETE, HEAD, OPTIONS, TRACE, and CONNECT. Information can be represented in multiple formats, such as JavaScript Object Notation (JSON) and XML.

Figure 1 graphically depicts the general functioning of a software architecture and the interrelationship between its components.

4. COUPLING SOFTWARE METRICS

A coupling software design is defined as the level of dependence between components; the more dependence an architecture has, the worse the design is.

An application that is built with strong coupling components raises the complexity of maintenance and does not allow scaling, thus its extension can be very costly in time and money.

Software coupling metrics are divided into two major groups: metrics for procedural programming and metrics for object-oriented programming [37–39]. Several metrics have been proposed by several researchers, such as Stevens, Myers, and Constantine [40] who introduce the concept of coupling in procedural programming and Dhama [42] proposes a metric that is focused on measuring the coupling of components individually. Martin Hitz and Behzad Montazeri [43] define coupling levels for object-oriented programming.

4.1. Fenton and Melton Metrics

Baker, Fenton, Gustafson, Melton and Whitty [41] propose to measure the degree of coupling between two components of the system x and y, by Eq. (1).

$$C(x,y) = i + \frac{n}{(n+1)}, \quad (1)$$

where: n is the number of interconnections between x and y, and i is the level of the highest (worst) coupling type found between x and y.

Table 1. Coupling Types Defined By Baker, Fenton, Gustafson, Melton And Whitty [41]

Coupling Type	Coupling level	Definition of coupling
Content	5	The x component refers to internal data or modifies a statement in the component y.
Common	4	The x and y components refer to the same global data area.
Control	3	The x component passes a control parameter to the component y.
Stamp	2	The x component passes a record type as variable to the component y.
Data	1	The x component passes a parameter with simple data or a homogeneous data structure and does not incorporate a control element.

The level of coupling is based on Stevens, Myers, and Constantine's classification [40].

On the basis of the abovementioned metrics, the following observations can be made:

i. Baker, Fenton, Gustafson, Melton and Whitty's metric [41] is a quantification of Stevens, Myers, and Constantine's coupling levels [40], whereas

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Dhama's metric [42] considers the number of variables or parameters belonging to categories that are less directly influenced by Stevens, Myers, and Constantine's classification [40].

ii. The highest coupling level between two components is the principal determinant of the coupling value in Baker, Fenton, Gustafson, Melton and Whitty's [41] metric. The coupling value increases as the number of interconnections between the two components increases.

iii. Baker, Fenton, Gustafson, Melton and Whitty's metric [41] considers all types of interconnections between components. It considers the same complexities and the same effects on the coupling.

iv. Dhama's metric [42] considers the coupling effect of a parameter that is the same as the effect of a global variable, which is based on Stevens, Myers, and Constantine's classification scheme [40].

In this work, the Baker, Fenton, Gustafson, Melton, and Whitty [41] coupling measure has been adopted.

5. SOFTWARE DEVELOPMENT USING A DECOUPLING, LAYERED AND WEB RESTFUL SERVICES APPROACH

An information system is a software system that helps to perform a specific function. Its main objective is to capture, process, and obtain information from a business. Generally, it is tailored to the needs of a company.

These systems are dynamic in time, that is why it is suggested to develop them with software decoupling techniques, in layers and using web services; so that the maintenance cost is minimal, and the system can be scalable and extensible.

Layered development can be physical or logical. The physical layered development refers to the server layer, while the logical layered development refers to software. However, the physical layered development depends on the logical layered development, which is the fundamental part to be able to decouple physical servers. Hereafter is the logical layered development will be discussed. The main idea of developing applications in layers is to decouple the data access layer, the business layer, and the presentation layer.

Layered development is a form of software decoupling. However, another form of decoupling is inherent in programming languages, such as C# y JAVA.

Mainly, two major types of decoupling exist: (1) decoupling the code through interfaces, specifying only the signatures of the methods that are called contracts, but not their implementation. This helps to abstract how it is programmed internally, that is, it is necessary to know only what data come in and what result can be expected. (2) the inversion of control of dependency injection, which consists in creating objects of classes at runtime without being instantiated inside the code. This mechanism provides an application server instance with an object and injects it in the code at runtime, without needing to know which class is instantiated, but until the moment it is required at runtime.

Finally, the Service Oriented Architecture (SOA) is another form of software decoupling. It currently uses several systems integration technologies, such as the JMS, Web Services, and the Web RESTful. The advantage of using the Web RESTful is that it is a stateless technology and is lighter in the network transport layer, which makes it a very efficient technology for data access in information systems.

Journal of Theoretical and Applied Information Technology
15th April 2020. Vol.98. No 07

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

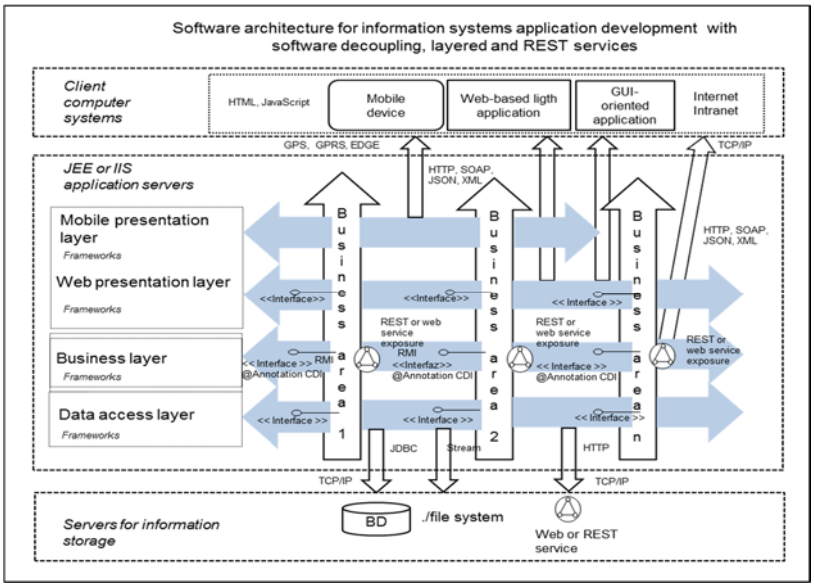

Figure 1: Software Architecture For Information Systems Using Software Decoupling, Layered, And Web Restful Services. Horizontally, The Frameworks That Help To Develop A Software Application Are Represented. Vertically, An Organization's Areas, Which Consist In The Data Layer, The Business Layer, And The Presentation Layer, Are Listed.

5.1. Layered Approach In Software Development

In the literature, it is common to find the definition of layer as physical layer and the definition of tier as logical layer. In a layered approach in software development, the expression 'logical layers' is used. The first division of layers that must be done in a web application includes data access and business logic, also called the backend. In addition, the user interface (visual part) is called the frontend. As the first layer, the backend includes the data layer that can come from several sources, such as a database server, a file system, a web service, or a combination of these data sources.

The second layer contains Data Transfer Objects (DTOs) that are generally Plain Old Java Objects (POJOs), which are logical representations in Java objects of database tables. Data Access Objects (DAOs) are methods that contain data and convert them into more complex objects, such as beans or lists of objects.

Finally, the third layer is the presentation layer, is where the user utilises a system's interface, as a browser or a mobile application to display information. In this layer, different technologies are available, for example, JSF or Java Server Pages (JSP) can be used.

5.2. Web Services

A web service is a software component that is identified by a URI, its interface, parameters or attributes, and can be described by the XML standard format. Web services can be accessible from the Internet or Intranet and may need or not a browser. This means that a web service can be called from an information system without human intervention. Web Services are developed in a wide variety of programming languages, such as C++, Java, Visual Basic, and others. They are independent of the platforms and the languages, and use protocols like HTTP, FTP, SOAP, CORBA, COM, DCOM, IIOP, SMTP, and RPC; they are oriented to the services architecture [44].

5.3. RESTful Web Service

Roy Thomas Fielding et al. [45] created the REST architecture in 1999, he later coined the term in 2000 in his doctoral thesis on the web. The REST uses the HTTP protocol to obtain data using a URI. The REST contains the principles of software architecture. The RESTful consists of web services that follow the REST principles. The advantages of the REST [45] are separation of resources, representation, visibility, security, scalability, and performance.

5.4. JavaScript Object Notation

JSON is a text-mode data exchange format derived from the JavaScript language [46]. Depending on the data type to be transferred, it may

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

be more appropriate to use XML or JSON. Both serve to the same purpose; the difference is that JSON uses less metadata in the HTTP header than XML, which is carried over SOAP and HTTP. JSON supports only two types of data, objects, and arrays [46].

6. METHOD AND MATERIALS

This section details the experimental design, the methodology for evaluating information systems, and the hardware and the software that were used.

6.1. Designing Two Information Systems Using Coupling And Decoupling Techniques

The objective of this experiment was to demonstrate that the use of decoupling techniques and tools for the design of information systems helps to optimise its functionality and maintenance, compared to the development of information systems that do not use these techniques. For the development of these information systems, the same hardware and software tools were used, so that both systems had the same test conditions.

6.2. Experimental Design

Once all the software for the systems development was installed (see annexe), a data model was created for both systems. The data model was executed in Oracle Database. The same data model was also executed in PostgreSQL. For this experiment, it was chosen to reduce the data model in four tables: Employee, Salary, Department, and Payroll. Subsequently, a coupling code was developed including the reading of the Employee table in a single class. Using a connection string to the database statically, the native query is injected to the database engine, which obtains the data as a response and is displayed by *out.println* in HTML.

Then, the same reading of the Employee table for the decoupled system was developed by using the Java Persistence API (JPA) framework to data access, the Enterprise JavaBeans (EJB) framework to the business layer, and the Java Server Faces (JSF) framework to visualize the data. All code was written on Eclipse Neon IDE for Java; both systems were deployed and executed on the application server WildFly 10 which implements the JEE specification.

For this experiment, quantitative tests were done to collect measurements. The quantitative measures for both systems included: (1) the time to execute a single query; (2) the response time simulating user load by making requests repeatedly to the system in a synchronous way; (3) the size of the traffic generated in the network. The validity of recorded data is made for both applications with load and without load.

Measurements of response time, concurrence simulating load, and network traffic are directly obtained from the systems execution. The weighting for each of the characteristics to be measured is a range of integer values from 0 to 2. 0 means that the application does not have this characteristic or that is deficient. 1 means that the characteristic is sufficient. 2 means the characteristic is excellent. In the end, measures from both systems were added. The system that obtains the highest score is the one that shows the best characteristics and, therefore, the one that will be recommended to design information systems in this work.

6.3. Methodology

The methodology included the design of two systems, one system was developed with coupling techniques and the other with decoupling techniques. Both systems accessed to the same data source. The methodology included:

- Using Java programming language to build both systems.
- Collecting measures of the behaviour of both systems.
- Comparing and presenting the results of the two systems.
- Plotting results.

Figure 2 shows the proposed server schema for this research.

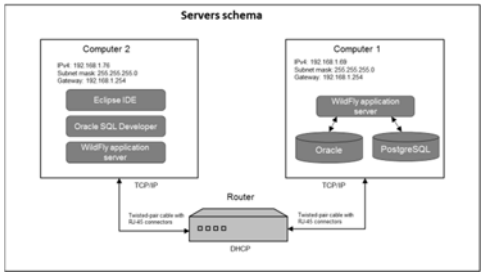

Figure 2: Server schema

7. TESTING AND RESULTS

Using a decoupling approach for information systems, with techniques such as the creation of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

software components in layers, their connection at runtime, and the use of RESTful services on networks, optimises the system's functionality and maintenance.

To verify this, two systems were developed in Java, with the same hardware and software platform, and had the same data source. One application was created without code decoupling techniques and the other with code decoupling techniques. Eight tests were designed to evaluate:

1. Coupling code vs. decoupling code using frameworks.
2. Using a database connection with a static string vs. a decoupled connection using a Java Naming and Directory Interface (JNDI) component.
3. Calling a coupled method without using an interface vs. calling decoupling code by creating an interface.
4. Calling a method by coupled code vs. calling a method using dependency injection.
5. Coupling code test vs. using a web RESTful service.
6. Creating a coupled code vs. creating a native query with Java Persistence Query Language (JPQL) from JPA.
7. Test execution using threads, for both, the coupled code, and the native query with JPA JPQL, removing injection of EJB, in the business layer.
8. Testing without decoupled physical servers.

The quantitative characteristics were the response time to execute each application (taking the response time from the request is made until the result is returned to the user), concurrence simulating load (users making reading requests to the database), and network traffic.

To obtain the execution time measures for each test, only one file was written to disk, recording the start time and the end time, as presented in Table 2.

Table 2: Runtime Without Load

	Time and Start Date	Time and End Date	Total time in seconds, milliseconds
Coupling application			
First register	13:37:28.485 08/01/2017	13:37:29.379 08/01/2017	0.894
Second register	13:59:13.008 08/01/2017	13:59:13.737 08/01/2017	0.729
Third register	14:06:39.471 08/01/2017	14:06:40.260 08/01/2017	0.789
Average execution time in milliseconds			0.804
Decoupling application			
First register	13:52:48.139 08/01/2017	13:52:49.002 08/01/2017	0.863
Second register	14:11:08.523 08/01/2017	14:11:09.285 08/01/2017	0.762
Third register	14:24:59.838 08/01/2017	14:25:00.528 08/01/2017	0.690
Average execution time in milliseconds			0.771

7.1. Coupling Calculations Using Fenton And Melton's Metrics

Table 3 shows coupling calculations based on Baker, Fenton, Gustafson, Melton and Whitty's work [41], as Eq. (1) indicates.

Table 3. Coupling Calculations Based On Baker, Fenton, Gustafson, Melton And Whitty's Work [41].

Coupling Type	Value
Content	5
Common	4
Control	3
Stamp	2
Data	1
No coupling	0

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

For the coupled code, n = 1, because it is only one module that does not make calls to other modules in different layers. i = 5, since it is a coupling code by content. The result of computing Eq. (1) with the above values is 5.5

$$\text{Coupling } C(x, y) = 5 + 1/(1+1) = 5.5$$

For the decoupled code, n = 3, because the data call is made from the presentation layer, passes through the business layer, and finally reaches the data layer. The value of i = 1, because only the entity name is passed, which is Employees table to bring information from the method where it is called. This results in Eq. (1) is

$$\text{Decoupling } C(x, y) = 1 + 3/(3+1) = 1.75$$

Therefore, the coupling measurement between the two codes is:

$$\text{Coupling } C(x, y) = 5 + 1/(1+1) = 5.5 > \text{Decoupling } C(x, y) = 1 + 3/(3+1) = 1.75$$

7.2. Recording Runtime Test Simulating Load

Table 4 shows the descriptive statistics and t-test results after reading 100 times the table Employee, which represents load simulation.

Table 4. Descriptive Statistics And T-test Results

Application	Media (seconds)	SD	T	Sig.
Coupling	0.5246	0.238	-45.90	4.21E-68*
Decoupling	2.842	0.264		

*p<0.001

The measures were collected by registering the *HTTPRequest* and *HTTPResponse* byte sizes; the NetTraffic application (a free tool) was used. The transfer rate of the coupled application has a minimum average value of 70 Kb/s and a maximum average value of 80 Kb/s as it is recorded by NetTraffic, since the native query that is injected into the database only extracts the data from the Employee table of the database relational model. The transfer rate in the decoupled system, there is a minimum average value of 120 Kb/s and a maximum average value of 200 Kb/s, that is, the object-oriented model of Java entities is declared on *FetchType.EAGER* persistence mode. This means that, although the operator only asks to retrieve data from the Employee table, it must also retrieve data from the Salary table and the Department table to meet its object-oriented model or mapping referential integrity in the database tables. Figure 3 compares the network transfer rate for the two systems.

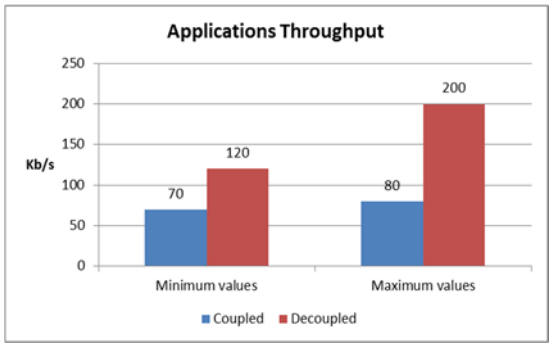

Figure 3: Number Of Bytes Of Httprequest In Kb/S.

The following experiment was conducted without accessing the database; only methods containing mathematical operations of addition, subtraction, multiplication and division were allowed. Figure 4 presents the results of this experiment; the horizontal axis shows the number of executions of the methods.

When the application server was started (executions 6, 11, 16, and 21), the coupled methods that were contained in a single class run faster than the decoupled methods that were declared in the classes that are created after the main class, which calls them (see Fig. 4). Once the objects have been created, in the following trials the execution time of the decoupled methods increases, and it even equals the time of the coupled methods.

However, as soon as the memory is used by other functions, the objects are lost, therefore the coupling method again surpasses the execution time of the decoupled method. This is related to memory usage. In general, the decoupling methods exceeded the execution time of the coupled methods.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

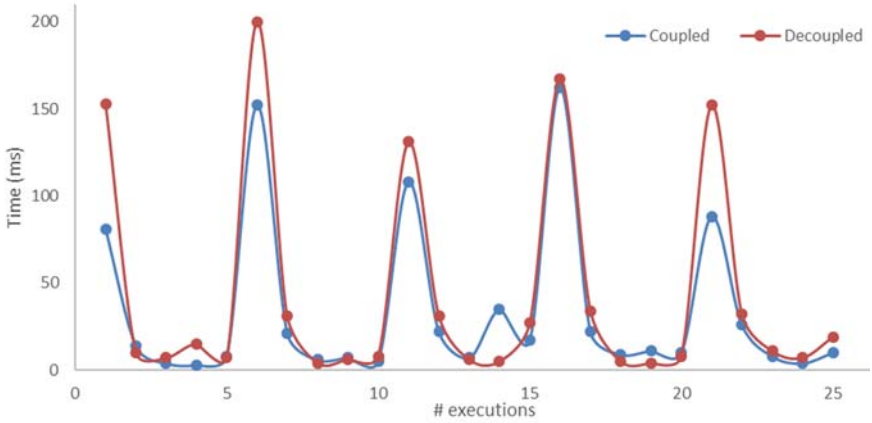

Figure 4: Execution Time Of Both Systems

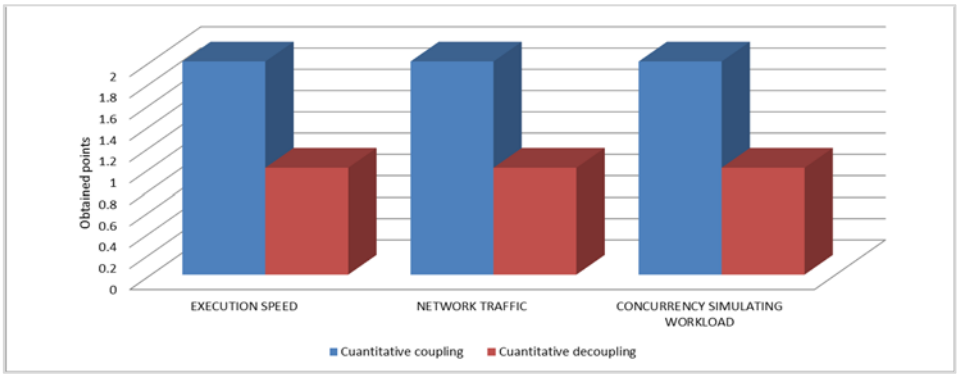

Figure 5: Comparing Characteristics Of Coupling Vs. Decoupling Software

Summarizing and adding the results of the three evaluated characteristics (i.e., execution time, load simulation, and network traffic), except test number 6 (i.e., creation of coupling by code vs. a native query with JPQL from JPA), the results are shown in Table 5.

Table 5. Summary Of Quantitative Values Between Systems E=Excellent, S=Sufficient, D=Deficient

Characteristic	Coupling Software System				Decoupling Software System			
	E (2)	S (1)	D (0)	Subtotal	E (2)	S (1)	D (0)	Subtotal
Execution time	*			2		*		1
Network traffic	*			2		*		1
Concurrency simulating	*			2		*		1
Total points				6				3

The final results show that the coupled application gains in execution time and the network traffic is lower than the decoupled software application. Figure 5 shows the results of this research.

8. CONCLUSIONS

The results of the study suggest that a coupling approach gives a higher speed of execution than a decoupling approach, since the latter goes through a large number of layers (frameworks), which use intermediate languages, and it consumes time in creating the necessary objects for its execution. The first time they are instantiated (in some cases injected) in the memory of the containers of objects and they are created, the speed improves, but it does not surpass the speed of the coupled approach. Only when native queries of each database handler are executed, the speed is exceeded, but the reuse of software components is lost. The size in network bytes is much greater, as a result of the number of objects that are created in the frameworks, because they generate the entire object model of tables in the database, respecting its referential integrity of the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

data model. In spite of this, it is worth sacrificing a little speed and size in the network for the versatility offered by the decoupled software components, their collaboration of group development, and its maintenance by the abstraction of the parts that integrate a software application

The use of RESTful services produces the maximum decoupling degree for a software application. RESTful services allow to create access to data in a distributed way, removing transaction loads to the main application servers, offering high scalability and extensibility, and complying with the SOA architecture.

RESTful services, by their design, are services based on stateless communications without any persistence. Therefore, it is necessary to implement methods of security and authentication of users in the system; tokens are used on the server side of applications. They travel over the HTTP protocol for both, the *HttpRequest* and *HttpResponse* for each user authenticated in the system.

In software systems, the JPA framework helps to control user data transactions in a simpler way, guaranteeing the integrity and security of the information. The contrary occurs in the coupled systems, where is responsibility of the programmer ensures the integrity and security of the company's database.

Likewise, the EJB framework implements the security of its own objects within the containers of JEE application servers. At the presentation level, the JSF framework implements the objects that can be accessed by some users' roles in the applications. Therefore, in the applications that are developed with frameworks, these provide the security, in a simple way.

Software decoupling, layers development-based frameworks and RESTful services proved to be very useful and valuable for developing information systems, specifically where the response time is not critical. Whereas software coupling, is convenient for applications where the response time is critical.

ACKNOWLEDGEMENTS

This work was supported by Instituto Politécnico Nacional (Grant SIP20201101 and SIP2020169).

REFERENCES

[1] Acosta-Gonzaga, E., Alvarez-Cedillo, J., and Gordillo-Mejía, A., Arquitecturas en n-Capas: Un sistema adaptivo, *Polibits*, 2006, vol. 34, pp.34–37. DOI: 10.17562/PB-34-7.

[2] Rojas-Rodríguez, M., Acosta-Gonzaga, E., and Gordillo-Mejía, A., Propuesta de un patrón arquitectónico para programación distribuida, *Revista Internacional de Sistemas Computacionales y Electrónico*, 2011, vol. 3, no. 5, pp. 40–50 Escuela Superior de Cómputo.

[3] Dijkstra, E.W., The Structure of the 'T.H.E.' mulitprogramming system, *Communications of the ACM*, 1968, vol. 18, no. 8, pp. 453–457.

[4] Parnas, D., On a 'Buzzword': Hierarchical structure, *Proc. Information Processing 74, IPIP Congress 74*, 1974, pp. 336–339. North Holland Publishing Company.

[5] Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice, 3rd ed.*, Upper Saddle River, NJ, USA: Addison-Wesley, 2013.

[6] Yang, QL., Li, JL., Xing, JC., Wang, P., and Wang, RH., Design of Software Architecture of Intelligent Information System of Protective Engineering, *Proc. 3rd International Conference on Computational Intelligence and Industrial Application (PACIIA2010)*, Wuhan, Peoples R. China, 2010, vol. VIII, pp. 224-228.

[7] Ryoo, I., Na, W. and Kim, S, Information exchange architecture based on software defined networking for cooperative intelligent transportation systems, *Cluster computing-the journal of networks software tools and applications*, 2015, vol. 18, no.2, pp. 771-782.

[8] Yang, QL., Li, JL., Xing, JC., Wang, P., and Wang, RH., Design of Software Architecture of Intelligent Information System of Protective Engineering, *Proc. International Conference on Intelligent Computation and Industrial Application (ICIA2011),* Hong Kong, Peoples R. China, Jun. 18-19, 2011, ICIA2011, vol. III, pp. 224-228.

[9] Nikkila, R., Seilonen, I., and Koskinen, K., Software architecture for farm management information systems in precision agriculture, *Computers and Electronics in Agriculture,2010,* vol. 70, no. 2, pp. 328-336.

[10] Dennis, E.H., and Mugisa, E.K., Reusable software architecture for an accounting

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

information system, *Proc. IASTED International Conference on Software Engineering, Innsbruck, Austria*, Feb. 17-19, 2004, pp. 275–280.

[11] He, G., Feng, C., Yuxin, W., and Yuanyuan, S. A reusable software architecture model for manufactory management information system, *Proc. 26th Annual International Computer Software and Applications Conference, Oxford, England*, August. 26-29, 2002, pp. 469–471.

[12] Singh, G. B. and Gobrogge, S., Information system architecture for developing reusable testplans for embedded software, *Microprocessors and Microsystems*, 2001, vol. 24, no. 9 pp. 453-461.

[13] Vogel-Heuser, B., Feldmann, S., Folmer, J., Rösch, S., Heinrich, R., Rostami, K., and Reussner, R., Architecture-based Assessment and Planning of Software Changes in Information and Automated Production Systems State of the Art and Open Issues, *IEEE International Conference On Systems, Man, And Cybernetics (SMC 2015): Big Data Analytics For Human-Centric Systems, Book Series: IEEE International Conference on Systems Man and Cybernetics Conference Proceedings,* 2015, pp. 687-694.

[14] Brunie, L., Coquil, D., and Simon, S., Software architectures for collaborative proxies in wide area information systems, *Proc. 12th International Conference on Database and Expert Systems Applications (DEXA), Tech. Univ. Munich, Munich, Germany*, September 3-7, 2001, pp. 146–150.

[15] Tarver, B., Christensen, E. and Miller, A., The Wireless Information Transfer System (WITS) architecture for the Digital Modular Radio (DMR) Software Defined Radio (SDR), *Proc. IEEE 21st Century Military Communications Conference (MILCOM 2000)*, Los Angeles, CA, Oct. 22-25, 2000, pp. 226-230.

[16] Dorn, C., and Taylor, R.N., Coupling software architecture and human architecture for collaboration-aware system adaptation, *Proc. 35th International Conference on Software Engineering (ICSE, 2013), San Francisco, CA*, May 18-26, 2013, pp. 53–62.

[17] Guetat, S.B.A., and Dakhli, S.B.D., Building software solutions in an urbanized information system: The 5+1 software architecture model, *Procedia Technology*, *4th Conference on Enterprise Information Systems/Int Conference on Health and Social Care Information Systems and Technologies, Portugal*, October 3-5, 2012, , vol. 5, pp. 481–490.

[18] Guetat, S., and Dakhli, S.B.D., Software solutions construction according to information systems architecture principles, *Proc. Communications in Computer and Information Science, International Conference on Enterprise Information Systems, Vilamoura, Portugal*, 2011, pp. 408–417.

[19] Wessel, M. & Moeller, R. Flexible software architectures for ontology-based information systems, *Journal of Applied Logic*, 2009, vol. 7 no. 1, pp: 75-99.

[20] Kondo, Y. & Matsuo, M, Software defined architecture concept - a network system model for information networking architecture, *IEICE Transactions on Communications Electronics Information and Systems,* 1991, vol. 74, no. 11, pp. 3683-3693.

[21] Isenegger, D., Price, B., Wu, Y., Fischlin, A., Frei, U., Weibel R., and Allgöwer, B., IPODLAS - A Software Architecture for Coupling Temporal Simulation Systems, VR, and GIS, *ISPRS Journal of Photogrammetry and Remote Sensing,* 2005, vol. 60, no. 1, pp. 34-47. DOI:10.1016/j.isprsjprs.2005.10.003

[22] Newman, S., *Building Microservices: Designing Fine-Grained Systems*, USA: O'Reilly, 2015. 282 pages. ISBN-10: 1491950358, ISBN-13: 978-1491950357.

[23] Cai, Y., Feng, Q., Kazman, R., Mo, R., and Xiao, L., Decoupling level: A new metric for architectural maintenance complexity, *Proc. IEEE/ACM 38th International Conference on Software Engineering (ICSE),* 2016, pp. 499–510.

[24] Chowdhury, I., and Zulkernine, M., Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities, *Journal of Systems Architecture*, 2011, vol. 57, pp. 294–313.

[25] Candela, I., Bavota, G., Russo, B., and Oliveto, R., Using cohesion and coupling for software remodularization: Is it enough?, *ACM Transactions on Software Engineering and Methodology*, 2016, vol. 25, no. 3, pp. 1-28. DOI: 10.1145/2928268

[26] Gui, G., and Scott, P.D., New coupling and cohesion metrics for evaluation of software component reusability, *Proc. 9th IEEE*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

International Conference for Young Computer Scientists, *2008*, pp. 1181-1186. DOI: 10.1109/ICYCS.2008.270.

[27] Gui, G., and Scott, P.D., Measuring software component reusability by coupling and cohesion metrics, *Journal of Computers*, 2009, vol. 4, no. 9, pp. 797-805

[28] Bidve, V.S., and Sarasu, P., Tool for measuring coupling in object-oriented Java software, *International Journal of Interactive Multimedia and Artificial Intelligence*, 2016, vol. 4, no. 1, pp. 812-820

[29] Zhe, M., and Kerong, B., A Software Decoupling Partition Method Based on Interactive Genetic Algorithm, *Procedia Engineering*, 2011, vol. 15, pp. 2875–2879. DOI: /10.1016/j.proeng.2011.08.541.

[30] Leymann, F., Loose coupling and architectural implications, *Multiagent System Technologies, 15th German Conference, MATES 2017, Leipzig, Germany, August 23-26, 2017, Proceedings. ISSN 0302-9743. Lecture Notes in Artificial Intelligence,* ISBN 978-3-319-64797-5, DOI: /101007/978-3-319-64798-2.

[31] Boissier, O., Ciortea, A., Florea, A.M., and Zimmermann, A., Give agents some REST: A resource-oriented abstraction layer for Internet-scale agent environments, *Proc. Conference on Autonomous Agents and MultiAgent Systems AAMAS, Sao Pablo Brazil,* 2017.pp. 1502-1504

[32] Casado, M., Ghodsi, A., Koponen, T., Raghavan, B., Ratnasamy, S., and Shenker, S., Software-defined internet architecture: Decoupling architecture from infrastructure, *Proc. HotNets. ACM Workshop on Hot Topics in Networks, Redmond, Washington,* 2012, pp.43-48.

[33] Fielding, R.T., Architectural styles and the design of network-based software architectures, *Doctoral (Philosophy in Information and Computer Science) Dissertation*, California: University of California, Irvine, 2000.

[34] Reyes, A.M., and Acosta-Gonzaga, E., Análisis de Modelos de Componentes, *Polibits*, 2002, vol. 28, pp. 9–17.

[35] Sznajdleder, P., *Java a Fondo, Curso de Programación*, México: Alfaomega Grupo Editor Argentino, 2016.

[36] Gallus, J., and Williams, G., *OracleAs 10g R3: Build Java EE Applications* (Vol. I, Student Guide), USA: Oracle, 2007.

[37] O'Regan, G., *Introduction to Software Quality*, Ireland: Springer, 2014.

[38] Lee, R.Y., *Software Engineering: A Hands-On Approach*, USA: Atlantis Press, 2013.

[39] Chhabra, J.K., and Gupta, V. A., Survey of dynamic software metrics, *Journal of Computer Science and Technology*, 2010, vol. 25, no. 5, pp. 1016–1029. DOI: 10.1007/s11390-010-1080-9.

[40] Stevens, W. P., Myers, G. J., & Constantine, L. L., Structure design, *IBM Systems Journal*, 1974, vol. 13, pp. 115–139.

[41] Baker, A. L., Bieman, J. M., Fenton, N., Gustafson, D. A., Melton, A., & Whitty, R., A philosophy for software measurement. *Journal of Systems and Software*, 1990, vol. 4, no. 1, pp. 277-281

[42] Dhama, H., Quantitative models of cohesion and coupling in software, *Journal of Systems and Software*, 1995, vol. 29, no. 1, pp. 65–74.

[43] Hitz, M., and Montazeri, B., Measuring Coupling and Cohesion in Object-Oriented Systems, *In Proceedings of International Symposium on Applied Corporate Computing*. 1995, pp. 25-27.

[44] Munsinger, L., Patel, S., Stokol, G., and Williams, G., *Oracle University, Learn Oracle From Oracle. Oracle 10g: Build J2EE Applications* (Vol. II Student Guide), USA: Oracle, 2005.

[45] Fielding, R.T., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., *Hypertext Transfer Protocol -- HTTP/1.1*, (1999, January 09). Retrieved from https://www.ietf.org/rfc/rfc2616.txt

[46] Sznajdleder, P., *JEE 7 a Fondo, Diseño y Desarrollo de Aplicaciones Java Enterprise*, Argentina: Alfaomega Grupo Editor Argentino, 2015.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ANNEXE

Hardware Tools

The hardware components were:

1. Personal computer Dell Inspiron 3647 with technical features:
   - Processor: Intel Celeron CPU G1820 at 2.7GHz, 2 cores.
   - Memory RAM: 4 GB.
   - Network Card: Ethernet 1000Base-T 10/100/1000 Mbps.
   - Operating System: Windows 8.1 Simple Language 64 bits.
2. Personal computer HP SlimLine 260-002-LA with the following features:
   - Processor: Intel Celeron CPU J3060 at 1.6 GHz, 2 cores.
   - Memory RAM: 4 GB.
   - Network Card: Ethernet 1000Base-T 10/100/1000 Mbps.
   - Operating System: Windows 10 Home Simple Language 64-bit.
3. HG532e TELMEX residential modem, HCP protocol dynamic IP mapping.
4. LAN and WLAN (Ethernet and WIFI), twisted pair network cables with RJ-45 connectors.

Software Tools and Network Configuration

The software used for the development of the systems was:

- Database Manager:
  - PostgreSQL-9.6.1-1-win64-bigsql.
  - Oracle XE112 win64.
- Application Server:
  - WildFly-10.1.0. Final.
- Integrated Development Environment (IDE):
  - Eclipse-jee-neon-1a-win32-x86_64.
  - Sqldeveloper-4.1.3.20.78-x64.
- Java Project Management:
  - Apache-Maven-3.1.1.
  - Drivers such as Java libraries to connect databases.
  - Postgresql-9.3-1100.jdbc41.jar to connect WildFly to PostgreSQL.
  - Postgresql-9.3-1100.jdbc41.jar to connect Oracle SQL Developer to PostgresSQL.
  - Ojdbc7.jar to connect WildFly to Oracle supporting JDK 6, 7 y 8.
- Virtual machine and Java Development Kit:
  - jre-8u111-windows-x64.
  - jdk-7u40-windows-x64.

PostgreSQL is a PostgreSQL License. WildFly is a GNU Lesser General Public License (GNU LGPL). Eclipse Neon is an Eclipse Public License (EPL). Apache Maven is an Apache License. All of them are Software Open Source. Oracle and Oracle SQL Developer are an Oracle License. For this experiment, Oracle tools were used for didactic purposes.

On computer 1 the following tools were installed:

- PostgreSQL-9.6.1-1-win64-bigsql and Oracle XE112 win64 databases.
- Java Runtime Environment jre-8u111-windows-x64, which is Java Virtual Machine and JDK Java Development Kit jdk-7u40-windows-x64, which are Java libraries.
- A WildFly-10.1.0 application server.
- Final win64 to run RESTful services.

On computer 2 the following tools were nstalled:

- Java Runtime Environment jre-8u111-windows-x64, which includes Java Virtual Machine and JDK Java Development Kit jdk-7u40-windows-x64, which are Java libraries.
- A WildFly-10.1.0 application server. End to run the main applications.
- IDE's Eclipse-jee-neon-1a-win32-x86_64 y Sqldeveloper-4.1.3.20.78-x64.
- Apache-maven-3.1.1.

Both databases were installed on computer 1 due to their technical requirements.