Toward a Software Architecture Design Method
A Framework for Architecture Design

Paola Yuritzy Reyes Delgado[1], Laura C. Rodríguez-Martínez[1], Hector A. Duran-Limon[2], José Manuel Mora Tavarez[3], González Ricardo Mendoza[1] and Mario Alberto Rodríguez Díaz[1]
[1]*Instituto Tecnológico de Aguascalientes, Av. Adolfo López Mateos No. 1801, Fracc. Bona Gens, Aguascalientes, Mexico*
[2]*Universidad de Guadalajara, CUCEA, Jalisco, Mexico*
[3]*Universidad Autónoma de Aguascalientes, Aguascalientes, Mexico*

1 STAGE OF THE RESEARCH

Software Architecture (SA) design is an essential activity in the development of software systems. According to Kim and Garlan (2010, p. 1216), a strong progress has been reached in software architecture design given that: "today we find growing use of standards, architecture-based development methods, and handbooks for architectural design and documentation".

However, due to the increasing complexity of software systems (Aleti et al., 2013), as well as the, "emerging trends in SA, i.e., Service oriented architecture, Product line architecture, Aspect oriented architecture, and Model driven architecture" (Qureshi et al., 2013), software architecture designs methods, are relevant to be elaborated, studied, documented, used and evaluated.

According to Rodríguez et al. (2012, p. 1), "Service-Oriented Software Engineering (SoSE) is a new paradigm of software engineering, which is focused on the design and implementation of service-oriented software systems (SoSS)".
Within this context, we conducted a literature review, in which we found that there are not any software architecture design methods specifically focused on the development of Web-based service-oriented software systems.

The main objective of this research is to define a software architecture design method for Web-based service-oriented software systems.

To accomplish this, the research model presented in Figure 1 is proposed, which shows the elements to be taken into account for the design of our method. Such elements are the following:

P1. Search and Study Information related to Our Research: This step investigates the background and context of the research problem or need. It identifies and studies the foundations and related studies.

P2. Analysis of Area and sub Areas of Knowledge, and related Studies: This step involves studying those design factors (if any) that are deemed to substantially affect the results when applying the method. This step also includes analyzing and classifying relevant information related to our research.

P3. Development of Conceptual Framework based on: Systematic Review, architectures and styles following the guidelines used in the Design Science research (Hevner, March, Park & Ram, 2004): In this step, we analize the data obtained from the conceptual framework.

P4. Method Development: Here, we construct the design method based on the analysis results of the conceptual framework.

P5. Presentation Method to Consultants and experts for!: (i) Firstly, the developed method is presented to a panel of experts for their evaluation. (ii) Secondly, we will carry out a logical argument validation, which "consists in the development of logical arguments, ! evaluate the proposed method using a case example and compare its effectiveness against others software architecture design methods. A group of students of software engineering will use our method and other methods to guide the development of the case example. Then, we will quantitatively evaluate the results obtained by using the different methods.

In the case of steps P1, …, P6 of Figure 1, the conceptual research method by Mora (2004) is used. Our current research progress is up to step P3. We are currently addressing step P4.

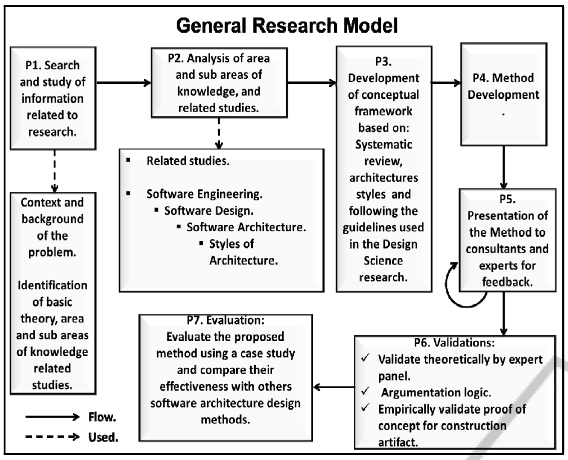

General Research Model
P1. Search and study of information related to research.
P2. Analysis of area and sub areas of knowledge, and related studies.
P3. Development of conceptual framework based on: Systematic review, architectures styles and following the guidelines used in the Design Science research.
P4. Method Development .
Related studies.
Software Engineering.
Software Design.
Software Architecture.
Styles of Architecture.
Context and background of the problem.
Identification of basic theory, area and sub areas of knowledge related studies.
P5. Presentation of the Method to consultants and experts for feedback.
P7. Evaluation: Evaluate the proposed method using a case study and compare their effectiveness with others software architecture design methods.
P6. Validations:
✓ Validate theoretically by expert panel.
✓ Argumentation logic.
✓ Empirically validate proof of concept for construction artifact.
Flow.
Used.

Figure 1: Research Model for the generation and validation of a software architecture design method for Web-based service-oriented software systems.

2 OUTLINE OF OBJECTIVES

The main objective of this research is to "define a software architecture design method for Web-based service-oriented software systems." The specific objectives are:

O1. Find the main contributions and limitations of current software architecture design methods.

O.2. Identify useful design principles for the software architecture design of Web-based service-oriented software systems.

O.3. Design the architectural development method based on the identification of activities and products of architectural design methods analysed in the specific objective O.1. Also, take into account the principles found in objective O.2.

O.4. Carry out a theoretical validation of our proposed software architecture design method by using logic argumentation as well as an assessment of an expert panel.

O.5. Carry out an empirical evaluation of our software architecture design method by means of a case example.

O.6. Compare the effectiveness of our software architecture design method with others software architecture design methods. The effectiveness of software design method will be tested by a group of software engineering students. The evaluation process will be twofold. Firstly, we will measure to which extent an architecture design produced by a software design method satisfies the functional and not functional requirements. Secondly, we will measure to which extent the risks associated with the construction of a software system are reduced (Pressman, 2002).

3 RESEARCH PROBLEM

This research problem is located into the knowledge area of Software Design (SWEBOK, 2004), within Software Engineering.

According to Gu and Lago (2009, p. 289), "It is often argued, in both academia and industry, whether the existing software engineering and architecting approaches (including techniques, methods and tools) are applicable as is in the context of SOA [Service Oriented Architecture]. However, since their 'real' differences with TSE [Traditional Software Engineering] remain fuzzy". "As a result, new approaches and design principles to build service-oriented systems have been continuously emerging".

Within this context, we conducted a literature review, in which we found that most of software architecture design methods are focused on the development of generic software systems, i.e., those methods do not consider the application domain area. A disadvantage of using generic methods is that it is not possible to emphasize the characteristics and specific goals of the application domain. For example, "architectural design of information systems emphasizes dates modeling, and architecture design of telecommunication software emphasizes continuous operation, live upgrade, and interoperability" (Hofmeister et al., 2007, p. 106). Only a few software architecture design methods are focused on different domains for which they were created (types of systems, type of company: e.g., large or small, etc.) (Hofmeister et al., 2007). However, no efforts have been carried out for developing a software architecture design method, specifically, for Web-based service-oriented software systems.

4 STATE OF THE ART

In this section we first introduce some common terminology of the areas that are related to our research work. Following, we present some relevant related work.

The IEEE Computer Society defines **Software Engineering** (SE) as "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).” (SWEBOK, 2004, p. 1-9). According to SEWBOK (2004), Software Engineering (SE) can be divided into ten areas of knowledge: Software Requirements, Software Design, Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, Software Engineering Tools and Methods, and Software Quality. Other definitions include the provided one by Pressman (2002, p. xxix), which defines software engineering as “…the practical application of scientific knowledge in the design and construction of computer programs and associated documentation required to develop, operate and maintain.”

In the SE research area, **Software Development Methodologies** (SDMs) have evolved toward better methodologies (Vavpotic and Vasilecas, 2011). In the last four decades, a large variety of SDMs have been proposed (Avison and Fitzgerald, 2003). The evolution of such SDMs has been carried out in four areas (Rodríguez et al., 2008): pre-methodologies, rigor-oriented methodologies, agile-oriented methodologies and emergent service-oriented methodologies. Also, the SDMs can be classified according to different criteria, e.g., means the user's software needs in a functional or non-functional requirements or means the complexity size of the systems. Software system projects can be small, medium or large according to the demanded number of lines of code or required human effort (Rizwan Jameel Qureshi, 2012). Such projects can be also simple or complex: "from small and simple: e.g., software for simple web-shops; to highly specialized and complex software systems: software for information systems used in manufacture based on complex mathematical models" (Vavpotic and Vasilecas, 2011, p. 107).

All SDMs include an important task called **Software Design.** Such a task is defined by the IEEE 610.12-90 as: "the process of defining the architecture, components, interfaces, and other characteristics of a system or component” and “the result of [that] process” (SWEBOK, 2004, pp. 1-4). Also, the SWEBOK (2004) states that: "Viewed as a process, software design (the result) must describe the software architecture—that is, how software is decomposed and organized into components—and the interfaces between those components. It must also describe the components at a level of detail that enable their construction". At the same time, according to the IEEE 12207.0-96. standard, the **Software Architecture Design** and the **Detailed Design** are two core activities among the initial requirements analysis and final software construction (Brown et al., 1998; SWEBOK, 2004; Vogel, Arnold, Chughtai and Kehrer, 2011) as follow: “Software architectural design (also known as top level design, high-level design, macro-architecture): describing software’s top-level structure and organization and identifying the various components.” and “Software detailed design (also known as bottom level design, low-level design, and micro-architecture): describing each component sufficiently to allow for its construction.”

The concept of **Software Architecture** is used in various contexts (Mark et al., 2004), and there are numerous definitions of the term “software architecture” in Information Technology (IT), hence, it is a challenge to find one universal definition (Vogel et al., 2011). In this paper, we use the software architecture definition of ISO/ IEC/IEEE 42010:2011 (2011, p. 2) standard: "Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution". Then we use the term **Software Architecture Design Methods** to refer to the methods that “describe” how to design a software architecture. According to Vogel et al. (2011, p. 319) an architecture method is a “… prerequisite for a successful architecture".

In recent years, several design methodologies and frameworks have been proposed (Lee & Shirani, 2004; Hofmeister et al., 2007). We have reviewed four well-known software development methodologies as well as an emergent methodology. The main goal of this review was to identify and compare the software architecture design methods of such software development methodologies. Our analysis was based on the evaluation of software architectural design method carried out by Hofmeister et al. (2007).

We considered the Software Architecture Design Methods that are (explicitly or implicitly) included in five well-known Software Development Methodologies (SDMs).

1. Model-Based (System) Architecting and Software Engineering (MBASE), (MBASE, 2000, 2003)
2. IBM Rational Unified Process for Systems Z (Cantor, 2003; Péraire et al., 2007).
3. Unified Process for Education (UPEDU) (Robillard et al., 2004, 2012).
4. Team Software Process (TSP) (Humphrey, 1998; Donald, 2000; Humphrey et al., 2010).
5. Service-oriented Software Development Methodology (SoSDM) (Rodríguez et al.,

2009).

Our findings have been reported elsewhere (Reyes et al., 2013, under review) and suggest that: (i) software architecture design methods are not standardized; (ii) SA design methods share common goals but carry out different activities; and (iii) further empirical research is required to consolidate valuable knowledge gained from conceptual research in software architecture design methods.

Regarding the application domain area we will focus on for constructing our method, we have that: "Web services are self-contained, Web-enabled applications capable not only of performing business activities on their own, but also possessing the ability to engage other Web services in order to complete higher-order business transactions"(Yang, 2003, p. 35). In addition, according to Rodríguez et al., (2012, p. 5). "A service-oriented software system (SoSS) refers to a distributed and loosely-coupled software system which is constructed based on the definition and implementation of a suite of services that forms it".

A typical SoSS involves the features shown in the Table 1:

Table 1: Comparative Table of Object-Oriented Software Engineering (OOSE), Component-Based Software Engineering (CBSE) y Service-Oriented Software Engineering (SOSE) paradigms (Rodríguez et al., 2004).

Attribute	Object-oriented Software System	Component-based Software System	Service-oriented Software Systems
Key analysis entity	Class	Business component	Business service
Key design entity	Object (conceptual)	Component (conceptual)	Business computing service
Key building entity	Object (local runtime)	Component (local or distributed runtime)	ICT computing service
Coupling level with remainder software	Tightly	Medium	Loosely
Cohesion level	Normal	High	Very high
Platform Interoperability	Minimal or null	High	Very high
Typical technology	C++	JavaBeans	Web services from several languages (Java, C#, PHP)

5 METHODOLOGY

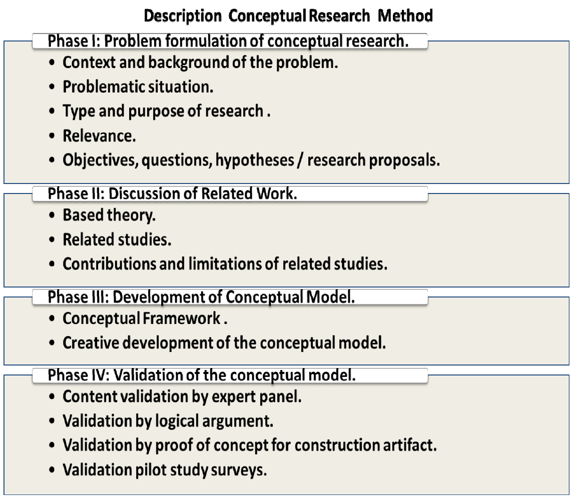
The research focuses on the study and creation of innovative model or artefacts, using two research approaches:

- Theoretical research (conceptual-analysis).
- Engineering research (through design- science).

The main purpose of this research is proposing a Software Architecture Design Method, whose design will be based first under the theoretical concepts included in General Design Methods, in order to obtain an applicable Method for develop Web-based service-oriented software systems, therefore the research classified as conceptual.

The conceptual research can be considered: "as the main source of generation of new theories, models or conceptual schemes - to complement the scientific cycle - then should be tested empirically or deductively using other research methods" (Mora, 2004).

"However, according a review of international literature in the field, has not been reported consensed phases to follow in the Conceptual Method" (Mora, 2004, p. 2). Therefore Mora (2004), proposes four phases to the conceptual method, which are based on general suggestions reported by researchers reffering to validating conceptual models, principles for evaluating theoretical research, and methods and techniques used in the qualitative assessment. Figure 2 shows the phases and tasks of the conceptual reseach method.


Description Conceptual Research Method

Phase I: Problem formulation of conceptual research.
- Context and background of the problem.
- Problematic situation.
- Type and purpose of research .
- Relevance.
- Objectives, questions, hypotheses / research proposals.

Phase II: Discussion of Related Work.
- Based theory.
- Related studies.
- Contributions and limitations of related studies.

Phase III: Development of Conceptual Model.
- Conceptual Framework .
- Creative development of the conceptual model.

Phase IV: Validation of the conceptual model.
- Content validation by expert panel.
- Validation by logical argument.
- Validation by proof of concept for construction artifact.
- Validation pilot study surveys.

Figure 2: Phases Description Conceptual Research Method.

This Conceptual Research Method suggests four forms of validation of the designed artifacts that are product of a research: (i) validation by experts' panel, (ii) validation by logical arguments, (iii) validation by proof of concept for artifact construction, and (iv) validation by pilot study.

"The studies of conceptual validation can be referred to the establishment of the degree to which the conceptual model satisfactorily meets the following criteria: (a) the conceptual model is supported by robust theories and principles, (b) the conceptual model is logically coherent and consistent with the studied reality and appropriate to the purpose for which the model was designed, and (c) the conceptual model brings something that is new and it is not duplication of another existing model" (Mora, 2004, pp. 9-10).

The directives or guidelines of design science proposed in Hevner, et al. (2004) will be followed to design the artifact that is product of the research: two paradigms characterize research in the discipline of Information Systems, the **Behavioral-Science** and **Design-Science**. "The Behavioral-Science paradigm seeks to develop and verify theories that explain or predict human or organizational behavior. The Design-Science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" (Hevner et al., 2004, p. 75). For this study, we consider just the Design-Science paradigm because we only seeks the creation, validation and evaluation of the architectural design method and does not predict or verify the behavior of individuals or organization with respect to the proposed architectural design method.

6 EXPECTED OUTCOME

The main contribution of this research aims to define a software architecture design method for designing the architecture of Web-based service-oriented software systems. Our method will be able to be applied in different national or international organizations.

As a future work we will be looking at using this method as a base framework to define an architectural style that we will be instantiated in several specific software systems architectures.

REFERENCES

Aleti, A., Buhnova, B., Grunske, L., & Koziolek A. 2013. Software Architecture Optimization Methods: A Systematic Literature Review. *IEEE Transactions On Software Engineering*, 39(5), 658-683.

Avison, D. & Fitzgerald, G. 2003. *Methodologies for Developing Information Systems: A Historical Perspective*. The Past and Future of Information Systems: 1976–2006 and Beyond, IFIP 19th World Computer Congress, TC-8, *Information System Stream*, August 21–23, 2006, Santiago, Chile.

Brown, W., Malveau, R., McCormick III, H., Mowbray, T. 1998. Anti Patterns – Refactoring Software Architectures, and Projects in Crisis, John Wiley & Sons, New York.

Cantor, M. .2003. Rational Unified Process for Systems Engineering. Rational Brand Services ,*IBM Software Group.*

Donald, M. 2000. The Team Software Process (TSP): An Overview and Preliminary Results of Using Disciplined Practices. (Tech. Rep. CMU/SEI-2000-TR-015), Carnegie Mellon, Software Engineering Institute.

Hevner, A., March, S., Park, J. & Ram, S., 2004. *Design Science in Information Systems Research*, Mis Quarterly, Vol. 28 No. 1, (pp 75-105).

Hofmeister, C., Kruchten, F., Nord, R., Obbink, H., Ran, A., & America, P. 2007. A general model of software architecture design derived from five industrial approaches. The Journal of System and Software, 80, 106-126, Elsevier Inc.

Humphrey, W. 1998. Three Dimensions of Process Improvement Part III: The Team Process, Software Engineering Institute.

Humphrey, W., Chick, T., Nichols,W., & Pomeroy-Huff, Marsha. 2010. Team Software Process (TSP) Body of Knowledge (BOK). (Tech. Rep. CMU/SEI-2010-TR-020). Software Engineering Institute, Carnegie Mellon University.

ISO/IEC/IEEE 42010:2011., 2011. Inernational Standar, Systems and software engineering - Architecture description ISO/IEC/IEEE 42010:2011, first edition, pp. 1-2. Retrieves Julio 26, 2013, from: http://www.iso-architecture.org/42010/index.html.

Kim, J., S., & Garlan, D. 2010. Analyzing architectural styles. *The Journal of Systems and Software, 83,* 1216–1235.

Kitchenham, B., 1996. Evaluating Software Engineering Methods and Tool, *Software Engineering Notes 21(1), (pp. 11-15).*

Lee, S., & Shirani, A. 2004. A component based methodology for Web application development, *The Journal of Systems and Software*, 71, 177-187.

Mark, W., Emery D., & Hilliard R. 2004. *ANSI/ IEEE Standard 1471 and System Engineering, System Engineering, Vol. 7, No. 3.*

MBASE. (2000). Guidelines for Model-Based (System) Architecting and Software Engineering (MBASE) 1997-2000, v2.2, 1-159, Center for Software Engineering, University of Southern California.

MBASE. (2003). Guidelines for Model-Based (System) Architecting and Software Engineering (MBASE) 1997-2002, v2.4., Center for Software Engineering, University of Southern California.Mora, J. 2004. *Descripción del Método de Investigación Conceptual,*

Technical Report No. 2, Department of Information Systems, Universidad Autónoma de Aguascalientes.
Péraire C., Edwards, M., Fernandes, A., Mancin, E., & Carroll, K. .2007. The IBM Rational Unified Process for System Z. IBM Rational Software.
Pressman, R. 2002. Ingeniería de Software – Un Enfoque Práctico, Quinta Edición, McGraw-Hill Práctica, 2002.
Gu, Q. & Lago, P., 2009. On Service-Oriented Architectural Concerns and Viewpoints*, WICSA/ ECSA. (pp. 289-299).*
Qureshi, N., Muhammad, U., & Ikram, N., 2013. Evidence in Software Architecture, a Systematic Literature. *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, (pp 97-106), ISBN: 978-1-4503-1848-8.
Reyes, P., Mora, J., Rodríguez, L., Duran-Limon, H., Mendoza R. & Rodríguez, M. (2013 under review). State of the Art of Software Architecture Design Methods used in Main Software Development Methodologies, *Encyclopedia of Information Science and Technology*, IGI Global.
Robillard, P. N., Kructen F., & d'Astous, P. 2004. Software Engineering Process with UPEDU, Adison Wesley.
Robillard, P. N., Kructen F., & d'Astous, P. 2011. Unified Process for Education , (UPEDU). École Polytechnique de Montréal, Retrieves October 27, 2013, from: http://www.upedu.org/.
Rodríguez, L. C., Mora, M., Vargas, M., O'Connor , R., & Alvarez, F. 2008. Process Models of SDLCs: Comparison and Evolution. M. Rahman Syed and S. Nessa Syed Editors, *Handbook of Research on Modern Systems Analysis and Design Technologies* and Applications, (p. 76-89).
Rodríguez, L., Mora, J., Álvarez, F., Garza, A., Durán, H., Muñoz, J. 2009. Diseño de un Modelo de Proceso de Ciclo de Vida de Sistemas de Software, en el Paradigma de Ingeniería de Software Orientada a Servicios. *Doctoral dissertation, Universidad Autónoma de Aguascalientes*.
Rodríguez-Martínez, L., Mora, M., Álvarez, F., Garza, L., Durán, H., & Muñoz, J. 2012. Review of Relevant System Development Life Cycles (SDLCs) in Service-Oriented Software Engineering (SoSE). Journal of Applied Research and Technology, 10(2), 94-113.
Rizwan Jameel Qureshi, M. 2012. Agile software development methodology for medium and large projects. *Journal IET Software*, Vol. 6, ISS.4, pp. 358–363.
SWEBOK. 2004. Guide to the Software Engineering Body of Knowledge, *IEEE Computer Society.*
Vavpotic, D. & Vasilecas, O. 2011. *An Approach for Assessment of Software Development Methodologies Suitability*, System Engineering, Computer Technology, ISSN 1392-1215.
Vogel, O., Arnold, I., Chughtai A., Kehrer T. (2011). Software Architecture, A Comprehensive Framework and Guide for Practitioners, Springer, ISBN 978-3-642-19735-2.
Yang, J., 2003. Web Service Componentization, *Communications of the ACM*, 46(10) (pp.35-40)