State of the Art of Software Architecture Design Methods Used in Main Software Development Methodologies

S

Reyes Delgado Paola Yuritzy
Instituto Tecnológico de Aguascalientes, Mexico

Mora Tavarez José Manuel
Universidad Autónoma de Aguascalientes, Mexico

Duran-Limon Hector Alejandro
Universidad de Guadalajara, Mexico

Rodríguez-Martínez Laura Cecilia
Instituto Tecnológico de Aguascalientes, Mexico

Mendoza González Ricardo
Instituto Tecnológico de Aguascalientes, Mexico

Rodríguez Díaz Mario Alberto
Instituto Tecnológico de Aguascalientes, Mexico

INTRODUCTION

Software Architecture (SA) design methods have been studied from the early 1990's decade given the increasing complexity and size in lines of code (LOC) of modern software systems (Garlan & Shaw, 1994). Nowadays, according to Kim & Garlan (2010), a mature area has been already reached given that: "*today we find growing use of standards, architecture-based development methods, and handbooks for architectural design and documentation" (*p. 1216).

However, due to the significant industrial demands toward software systems with an increasing complexity (e.g., new and multiple technologies development platforms and more technical and user-oriented demand of requirements) (Aleti, Buhnova, Grunske, Koziolek & Meedeniya, 2013), and due to the permanent need for efficient and effective software design process (Angelov, Grefen, & Greefhorst, 2011), the SA design methods are still a relevant research and academic area.

Software Architecture can be defined as: *"Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution"* (ISO/IEC/IEEE 42010, 2011, p. 2). The ISO/IEC 12207 standard reports that the Software Architecture Design and the Detailed Design activities are fundamental parts of the set of activities to be realized from Requirements Analysis until Software Construction (SWEBOK, 2004). SA design consists essentially in specifying its core components, the interrelationships among them, and a set of essential attributes expected for the final software. This is realized in a high level of abstraction (Weinreich & Buchgeher, 2012) and adequate SA design methods must be used. Therefore, missing this activity or a wrong conduction of it will have negative effects in further software development phases, and lately in the final software. Hence, SA design is a relevant activity for the development of software systems, and given the current complexity of modern software systems, it can be also considered a design challenge (Aleti et al., 2013).

There are a myriad of Software Development Methodologies (SDMs) since several decades ago (Rodriguez, Mora, Vargas, O'Connor & Alvarez, 2008; Vavpotic & Vasilecas, 2011). All of them provide either

DOI: 10.4018/978-1-4666-5888-2.ch724

an explicit or implicit guidance for a SA design. Nevertheless, their real utilization in the software development practice is scarcely reported (Chatzoglou, 1997; Fitzgerald, 1998). Furthermore, implicit and explicit SA design methods use a varied structure of activities and artifacts, as well as a particular non-standard nomenclature. This dual problem: the lack of utilization of SA design methods and the non-standardization of them precludes their correct application by software professionals.

Thus, in this article we review and compare the implicit and explicit SA design methods included in four well-known international Software Development Methodologies plus a recently reported SDM for service-oriented software engineering. These Software Development Methodologies are:

1. Model-Based (System) Architecting and Software Engineering (MBASE), (MBASE, 2000, 2003).
2. IBM Rational Unified Process for Systems Z (Cantor, 2003; Péraire, Edwards, Fernandes, Mancin, & Carroll, 2007).
3. Unified Process for Education (UPEDU) (Robillard, Kructen & d'Astous, 2004, 2012).
4. Team Software Process (TSP) (Humphrey, 1998; Humphrey, Chick, Nichols & Pomeroy-Huff, 2010; Donald, 2000).
5. Service-oriented Software Development Methodology (SoSDM) (Rodríguez et al., 2009a).

This article aims to help software engineering academicians and professionals with the provision of a compact but substantive descriptive-comparative guide on a set of main SA design methods. However, this article does not pursue to be a direct learning source for any of the SA design methods here presented. Space limitations preclude this aim. Interested readers are refereed to additional particular readings.

BACKGROUND

This section presents fundamental concepts of: Software Engineering, Software Development Methodologies, Software Design, Software Architecture, and Software Architecture Design Methods. A systematic literature review process (Kitchenham & Charters, 2007; Kitchenham et al., 2009) was conducted by reviewing ten top journals in the discipline of Software Engineering. These journals were: Information and Software Technology (IST), Journal of Systems and Software (JSS), Software Practice and Experience (SPE), IEEE Software (SW), IEEE Transactions on Software Engineering and Methodologies (TOSEM), IEEE Transactions on Software Engineering (TSE), ACM Transactions on the Web, Advances in Engineering Software, IEEE Transactions on Services Computing, and Journal of Systems Architecture. The keywords used were: *software development methodologies, software design, software architecture, software architecture design methods, MBASE, RUP, UPEDU, TSP* and *SoSE methodology*. The search for this review, carried out in three academic databases, produced 543 papers. The first and third co-author filtered them by reading both title and abstract. Those papers that were not directly relevant to the focus of out research were excluded. As a result, we obtained 28 articles. The second author did a random (about 5% of the total) evaluation for classifying the selected vs non-selected papers in which no disagreements were found, Finally, these 28 papers were completed with 18 additional research products (e.g., journals papers, conference proceedings, technical reports, books and specialized websites) referenced in the 28 core papers and/or suggested by all authors. The selection of journals was based on their recognized ranking status (Glass & Chen, 2005). This set of 46 papers served as knowledge source for deriving the presented foundations of this section.

Software Engineering

The IEEE Computer Society defines Software Engineering 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 the SEWBOK (2004), Software Engineering can be divided into ten areas of knowledge: (1) Software Requirements, (2) Software Design, (3) Software Construction, (4) Software Testing, (5) Software Maintenance, (6) Software Configuration Management, (7) Software Engineering Management, (8) Software Engineering Process, (9) Software Engi-