ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 11(21), DOI: 10.17485/ijst/2018/v11i21/124487, June 2018*

A Methodological Approach for Software Architecture Recovery

R. Martin Monroy *, Julio R. Ribon and Plinio Puello

Department of System Engineering, University of Cartagena, Colombia; mmonroyr@unicartagena.edu.co, jrodriguezr@unicartagena.edu.co, ppuellom@unicartagena.edu.co

Abstract

Background/Objectives: The software architecture recovery process is an activity that is included in different contexts. However, the methodological proposals to perform this process do not take into account the particular needs of the context in which it is developed. The objective of this work is to propose a methodology for Software Architecture Recovery, responding the specific needs of the context in which the need to recover the architecture of a software product is presented. **Methods**: The model was obtained after applying the pattern-matching technique in order to establish the common aspects to all the proposals identified in the literature review. **Findings**: The results of the evaluation of the methodological proposal reveal the usefulness when recovering architectures, since it allows focusing attention on the most relevant aspects of recovery for the specific context in which the process is performed. **Novelty**: The defined methodological proposal is a new way of performing architecture recovery processes, which achieves more relevant results to the context in which the needs arise.

Keywords: Architecture Recovery, Methodological Approach, Reverse Engineering, Software Architecture

1. Introduction

From the beginning, reverse engineering has been used as a software maintenance technique[1], however, throughout its evolution, it has been used to solve several problems which software engineering is facing, such as: the recovery of architectures and design patterns, the re-documentation of programs and databases, the identification of reusable assets, the definition of traceability between software artifacts, the identification of the impact of software product changes, the restructuring of existing systems, the renewal of interfaces of user, the migration towards new architectures and platforms among others[2]. This characteristic has caused in the fields of application of reverse engineering to be extended to different contexts, such as: Software production, computer security, forensic computing and education[3].

On the other hand, architecture recovery techniques and methods identified in the literature review have been defined only for situations that arise in the context of software production[4]. For example, there are techniques that are specialized in reconstructing system documentation[5–9], in the evolution[10,11] and analysis[12] of the product, or in the review of compliance[13,14], while other techniques are used to achieve one or more purposes at the same time[15,16]. However, in each of the contexts in which reverse engineering is applied, different situations arise to affect individuals with unlike and particular purposes, additionally not all contexts have the same resources[3].

For example, in a context like computer security, there are situations such as the analysis of malicious software, which are addressed by security experts, whose purpose is to understand the structure and behavior of this type of program to solve the representing risks. Under these circumstances, there are no artifacts such as source code, configuration files, data structures or documentation. In addition, the views that need to be recovered are more oriented to understanding the modular structures and the execution profiles than the structures themselves. Therefore, there is a need to define a methodological proposal that guides the process of architecture recovery, taking into account the interests of the participants and

*Author for correspondence

the specific characteristics of each context, in which the situation to be solved is presented.

The main contribution of this work is the definition of a methodology to recover software product architectures, taking into account the specific needs of the context in which this process is carried out. It is aimed at software engineers and participants in architecture recovery processes. This methodological proposal is used as a guide to recover the artifacts that describe the architecture of software products, taking into account the purposes within the context in which it is applied, its circumstances, its characteristics and the available resources, which contributes to the solution of the problem posed. Additionally, the results of the case study applied for the evaluation of the methodology are documented.

Following this, the methods and materials used for the development of the research are presented. Then the methodological proposal is explained, detailing the sustained axioms and the established process. Subsequently, the results of the case study that was applied to evaluate the proposed methodology are presented. Later, the results are analyzed and finally the conclusions are presented.

2. Materials and Methods

The investigation was carried out in two phases. In the first, a review of the literature was made applying the methodological approach established by[17] and a characterization model of the architecture recovery process was defined. The results of this phase were published in[4,18]. In the second phase, the analysis of the documentation identified in the previous phase was carried out, applying the technique of pattern matching[19] to establish the aspects common to all the proposals. As a result of this phase, the methodological proposal for recovering software architectures was obtained.

Subsequently, the methodological proposal was evaluated by applying to single case study (embedded) with two units of analysis[19], in two use scenarios. The first scenario concerns the software production, since it is the typical context for which all the proposals identified in the review of the literature have been defined, becoming a mandatory unit of analysis. The second scenario corresponds to the context of education, since it is one of the contexts of the use of reverse engineering, in which no specialized proposal was found in the literature review.

The design of the case study was based on its five components: the research question, propositions, the units of analysis, the logic to connecting data to propositions, and the criteria for interpreting the findings[19]. In the first instance, the question arises from the intention of the case study: to evaluate the methodological proposal, for that reason it is posed as follows: How does the methodological proposal contribute to the recovery of the architecture of a software product? The main proposition states that the proposed methodology guides the process of recovering the architecture of a software product, taking into account the particular characteristics of the situation surrounding the context in which the process is carried out.

The units of analysis were taken from the contexts of reverse engineering. The first is the production of software, and the second is the context of education. The main logical connection between the data and the propositions in this case study was carried out using the analysis technique called the Logic Model, which consists of observing the coincidence of the observed empirical events and the theoretical events[13]. This model of analysis was selected because the case study is carried out to evaluate the proposed methodology, demonstrating that the empirical events observed when using it coincide with the theoretical proposal presented. To argue the interpretation of the findings, the main strategy used was analysis based on theoretical propositions to make logical inferences based on the conceptual proposals identified in the literature review.

3. Results and Discussion

This section describes the methodological proposal for architecture recovery, explaining the elements that comprise it and the process that defines it. Then the results of the case study that was applied to evaluate the proposed methodology are presented. Finally, the analysis of the obtained results is made, and future works are proposed.

3.1 Methodological Proposal

The process for recovering architectural views is presented as a methodology, because it not only refers to the technical procedures used to achieve the objective but also shapes the diversity of the entire set of knowledge required to achieve it, thus establishing the four axioms that differentiate a methodology[20]. The proposal is called SAReM (Software Architecture Recovery Methodology);

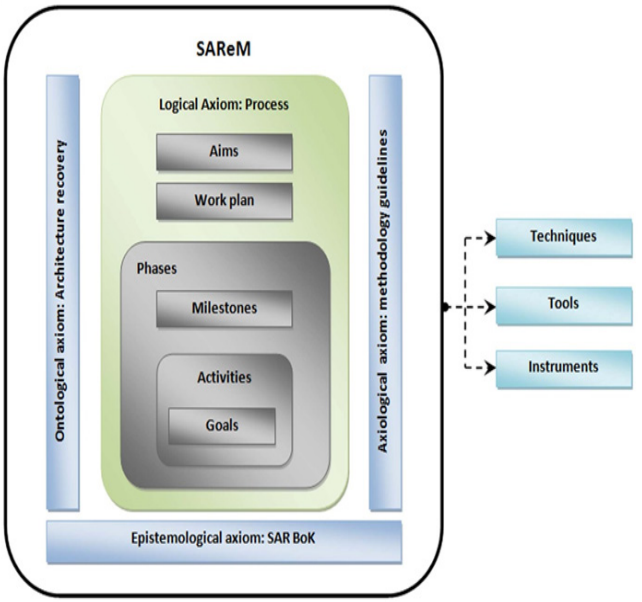

Figure 1. Methodological Approach Structure

the structure is represented in Figure 1, and each axiom is explained below. For the application of the methodology requires the use of techniques, tools and instruments to achieve the recovery of architectural views.

The epistemological axiom offers for clarity about what counts as knowledge and ways of knowing. It is constituted by Software Architecture Recovery Body of knowledge (SAR BoK), and is synthesized in the results of the literature review[18]. The ontological axiom defines the object of study and the nature of the reality that surrounds it. It is represented by the recovery of architectural views, as a need that presents itself in a situation under a specific context, as explained by[3].

The axiological axiom reveals what is counted as fundamental values and what consciousness is, establishing moral, ethical and normative judgments. It is represented by the following methodology guidelines: 1) the proposed methodology obeys a generic process that supports the recovery of architectural views taking into account the context in which the situation under study is presented. 2) The methodology does not define any new technique; it only integrates the existing ones so that their use is relevant to the context and the situation in which the problem arises. 3) The recovery process complies with the characterization model established by[4]. 4) The definition of the process specifies the activities that comprise it, indicating the logical sequence of its execution, the goals proposed; the resources received as input and those generated as output, as well as the instruments that they are used and the techniques applied to achieve the proposed goals.

Finally, the logical axiom indicates what is acceptable in terms of rigor and inference in the development of arguments, judgments, reflections or actions. It is constituted by the recovery process of the architecture that explains the development of the actions. The software architecture recovery process is executed in a specific context; it refers to the set of activities that are carried out under a logical and temporal order organized in phases, following a focus, and establishing one or several objectives. There are three types of approaches[21]: Top - down (from the abstract to the concrete), Bottom - up (from the concrete to the abstract) and hybrid.

The aim of the process establishes the target that is intended to be achieved. Some possible objectives that can be raised in an architecture recovery process are: documentation reconstruction, reusable assets identification, analysis of compliance, analysis for the identification of patterns, aspects, characteristics, roles, and collaborations, among others. An activity is a set of operations or tasks that a person or entity does, in which artifacts or pieces of information represented by documents or architectural elements are received, which are manipulated to meet the goals of the activity, with the help of resources represented by architecture recovery techniques and tools. The activities also operate with instruments, support elements, used to guide or record the activities, and the results obtained in each one of them.

An activity can consist of several sub-activities or tasks organized from a work plan; the latter is an instrument that establishes the sequence of activities to be carried out under a logical and temporal order, indicating those responsible, the resources to use, and the results that must be obtained. The logical order of the development of activities is defined from the realization of phases. A phase represents the states through which the process faces, depending on the compliance with the established milestones. The fulfillment of the goals of the activities contributes to the achievement of the milestones of the phases, and these in turn make possible the achievement of the aims proposed in the process.

3.2 The Process

The process is structured in four phases; each one has established one or several milestones and a set of activities that are carried out under a logical sequence described

in Figure 2. Each activity defines the goals that are to be achieved and the techniques and necessary tools to be performed. Similarly, for each activity the arguments justifying it (motivation), the stakeholders involved, the artifacts required as input and the artifacts that are generated as an output or result are identified. If the activity leads to the completion of tasks, the aforementioned aspects are specified for each task.

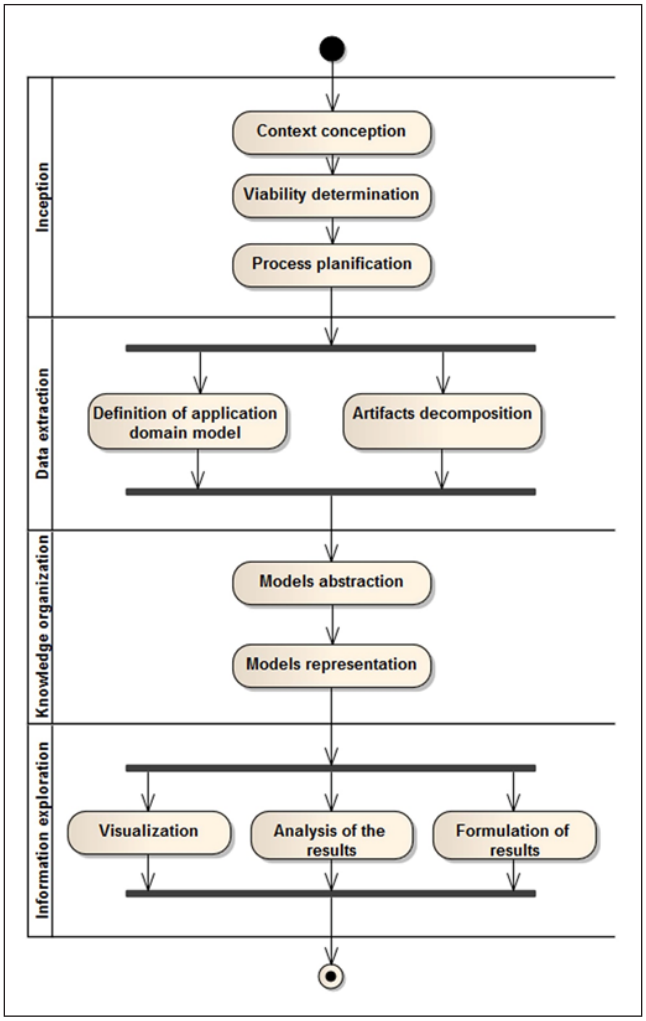

Figure 2. The Process

The order of execution of the activities that make up each of the phases is established in the work plan, depending on the analysis made of the context and the situation generated by the architecture recovery process. In the data extraction phase, the activity oriented to define the domain model of the application, appears if a Top-Down approach is used. In addition, the use of some architecture recovery tools, such as Imagix4D and the Moose platform, among others, merge the activities corresponding to the decomposition of artifacts, abstraction and model representation. No activity involved implies the modification of the product is, since this is a reverse engineering process. Each of the phases is summarized in Tables 1–4, specifying their milestones and activities. The goals and purposes for each activity, the techniques and instruments used, the inputs required and the outputs generated are presented.

3.3 Case Study

The results of the case study are presented for each unit of analysis. In the first instance, the analysis unit corresponding to the software-development process was carried out in a company of the industrial sector of the city of Cartagena. The stated objective is: Recover the software documentation, so that its functionality can be extended to new requirements requested by senior management. By applying the proposed methodology, it was possible to recover the following documentation for the ICR software product: 1) The general description of the product, 2) the domain model of the problem, 3) the domain model of the application and

Table 1. Inception Phase

Milestones	Specify the scope of the problem				
	Establish the feasibility of carrying out the process				
	Establish the work plan				
Activity	**Goals**	**Inputs**	**Techniques**	**Instruments**	**Outputs**
Define the context	Identify the context	Knowledge of the expert in reverse engineering	Inference	Characterization of the contexts of use	Description of the problem
	Pose a problem	Stakeholders knowledge and documentation	Interview	Template for the description of the problem	

Activity	Goals	Inputs	Techniques	Instruments	Outputs
Analyze the feasibility of the process	Define the feasibility of the process	Description of the problem, Stakeholders knowledge, documentation	Decision algorithm	List of architectural views according to context, checklists, characterization of Reverse Engineering tools	Decision about the viability of the process, The target views, List of available artifacts, List of required artifacts, Techniques to be used, Tools to be used, Cost of the process
Plan the process	Define the work plan	Description of the problem, target views, list of available and required artifacts, list of techniques and tools to be used	Resource allocation		Work plan

Table 2. Data Extraction Phase

Milestones	Identify the elements that make up the software product and the relationships that exist between them at a low level				
Activity	**Goals**	**Inputs**	**Techniques**	**Instruments**	**Outputs**
Define the application domain model	Define the domain of the application	knowledge of the expert, users and technical staff; and Description of the problem	Conceptual modeling	UML Diagrams	Domain application model
Decompose artifacts	Identify the elements that make up the system and its relationships	Software artifacts	Static analysis, dynamic analysis	KDM	System model in low level

Table 3. Knowledge Organization Phase

Milestone	Convert the system model to a low level in a high level model				
Activity	**Goals**	**Inputs**	**Techniques**	**Instruments**	**Outputs**
Models abstraction	Identify elements and relationships at a high level based on the low-level system model	Low-level system model, application domain model and expert knowledge	Manual, semi-automatic and automatic techniques	Mapping rules	Elements and relationships at a high level
Models representation	Represent software models at a high level	Elements and relationships at a high level	Mapping	UML, mapping rules	Model of the system in high level

Table 4. Information Exploration Phase

Milestone	Prepare the report with the analysis of the architectural views recovered, indicating the fulfillment of the objectives established for the process in the inception phase.				
Activity	**Goals**	**Inputs**	**Techniques**	**Instruments**	**Outputs**
Results visualization	Present the results in graphic form	Model of the system in high level	Metaphors, hyperlinks	UML	Target views represented in UML
Results analysis	Interpret the results obtained in the process	System model at a high level, domain model of the problem, application model and expert knowledge	Inference	Querying mechanism	Results analysis
Results formulations	Organize the results in a final report so that it can be interpreted by all stakeholders	Target views represented in UML, results analysis, domain model of the problem, application domain model, and expert knowledge	Drafting of the document		Process report

4) the system views: View of decomposition, layers view; classes view components and connectors view, and the assignment structures view.

In the data extraction phase, the domain of the problem and the domain of the application were defined. The elements that make up the system and its relationships were also identified. The system model in the low-level represented in XMI was obtained, fulfilling the milestone established in this phase. In the knowledge organization phase, mapping rules, manual and semiautomatic techniques were used to transform the system model into a low-level obtained in the previous phase, in the high-level system model. The milestone established in this phase was achieved, since the elements of the system, and their relationships were identified. Finally, in the information exploration phase, the results were presented in graphical form using models represented in UML, the interpretation and analysis of the results obtained in the process were made, and the final report presented to the Company of the industrial sector in the city of Cartagena was organized.

The Interpretation and analysis of the results, generated two recommendations for the Company of the industrial sector of the city of Cartagena: 1) to carry out a code cloning analysis to debug the system and identify possible reusable assets; and 2) to make a refactoring process, assigning more descriptive names to some modules and classes, defining interfaces for the modules responsible for the logic of the application, to guarantee the encapsulation, the reuse, facilitate the extensibility, and the maintenance capacity of the system.

The second unit of analysis corresponded to the context of Education. SAReM was used to support a teaching-learning process, in the development of an academic activity of the Object-Oriented Programming course, in which students of the systems engineering program of the University of Cartagena participated. The objective of the process was established: At the end of the class, students will be able to understand the concept of polymorphism and have the skills to use it in practical situations. For this unit of analysis, the results are measured not only from the recovered artifacts and the direct analysis made on the architectural views, but also the effect that the application of SAReM had on the apprentice, and the teacher's perception when using it.

Taking into account the characteristics of this context of use, it was only necessary to recover: 1) The general description of the product, 2) the domain model of the application and 3) the system views: View of decomposition, the view of classes and the view of components and connectors. The inception phase was carried out by the teacher, who defined the context of the problem, the viability of the process and the work plan. It also determined: The target views, the list of available artifacts, the list of required artifacts, the techniques and tools to be used and the cost of the process.

The teacher and the students participated in the data extraction phase. The teacher recovered the domain of the application, presented at the beginning of the academic activity so that the students understood the purpose of the software they analyze. Then the students recovered the view of classes using the Enterprise Architect tool. In the knowledge organization phase, only the teacher participated, who reconstructed the view of communication between processes through interfaces, using mapping rules, manual, and semiautomatic techniques. This view was used in the class to explain the students the structure of the software that was analyzed in the academic activity.

Finally, the exploration phase of information was carried out by students with the Enterprise Architect modeling tool. In this phase, the students visualized the results obtained in the process and with the help of the querying mechanism QModel-XMI[3] the interpretation and analysis of the results were made. This allowed them to identify possible polymorphic behaviors, which they reported as a result of the realization of the class activity, and which in turn were useful for the teacher to make the respective report on the learning activity supported by SAReM.

The use of SAReM in the academic process contributed to the learning objectives, because the results of the activity revealed that: 1) all students understood the concept of polymorphism and have the ability to identify where it is being applied. 2) The majority (75%) can apply the concept to modify the functionality of a software product.

3.4 Results Analysis

A new way of carrying out architecture recovery processes was defined, which complements existing proposals[9,10,12,14]. Unlike other methodologies, SAReM integrates relevant aspects to the specific needs of the context in which the architecture recovery process takes place, as shown in Table 5; where the results of the case study are

Table 5. Comparison of Analysis Units

Criterion	Analysis unit	
	Software Production Context	**Education context**
Aim	Extend the functionality of the software product.	Support the learning process
Objective	Recover software documentation	At the end of the class, students must understand the concept of polymorphism and have the skills to use it in practical situations
Situation	Company of the industrial sector has a software product that supports its production processes. It is necessary to extend the functionality of the system but the documentation is not available.	The study plan of the Systems Engineering program of the University of Cartagena has the subject Object-Oriented Programming. Polymorphism is a issue that must be addressed in this subject, therefore at the end of this course the student must understand and apply this concept.
Available resources	Software product source code, MARIADB relational database, database dictionary (Excel), business process models VPMN - Bizagi (production control, transport freight management, quality retention management), Persistence (xml of the databases it connects), Enterprise Architect. Expert in reverse engineering.	JHotDraw Java System Code, Student and Teacher Lab Guide, JHotDraw Documentation, Enterprise Architect, QModel-XMI, Reverse Engineering Expert.
Recovered documentation	The general description of the product, the domain model of the problem, the application domain model and the system views: Decomposition view, the layer view, the class view, the components and connectors view and the view of allocation structures.	The general description of the product, the application domain model and the system views: decomposition view, the classes view and components and connectors view.

compared for each unit of analysis. This allows defining a work plan according to the circumstances, the available resources and the purposes within the context in which the architecture recovery process is done. Consequently, it ensures that the process focuses on the most pertinent aspects of the context of the problem. Therefore, the results are relevant to specific needs. Thus, SAReM can be used in contexts such as software production, computer security, forensic computing and education.

The methodology corresponds to a generic process, which does not include activities related to the modification of software products, since its scope only includes the reverse engineering process. As future work, it is planned to apply the methodology in contexts related to computer security and forensic computing. It is also recommended to explore the possibility of extending this proposal to other fields of knowledge, such as mechanical and electronic engineering among others.

4. Conclusions

A methodology was defined to recover software architectures, based on the specific needs of the context in which the need to recover the architecture of a software product is presented. The results of the case study allow us to conclude that: 1) the methodology can be used to guide the recovery of the artifacts that describe the architecture of software products. 2) By applying this methodology, the attention can be focused on the most relevant aspects for the specific context in which the architecture recovery process takes place. 3) The use of this methodology allows the achievement of results more pertinent to the context in which the need arises. 4) To improve the methodology it is necessary to apply it in contexts such as computer security and forensic computing.

5. References

1. Ganesh S, Girish S, Arbind KG, Raghu N. FOCUS: An adaptation of a SWEBOK-based curriculum for industry requirements. 34th International Conference on Software Engineering (ICSE). 2012; p. 1215–24.
2. Canfora G, Di Penta M. Cerulo YL. Achievements and Challenges in Software Reverse Engineering, communications of the ACM. 2011; 54(4):142–51.
3. Monroy M, Arciniegas JL, Rodríguez JC. Characterization of the contexts of use of reverse engineering. Información tecnológica. 2017 July; 28(4):75–84.

4. Monroy M, Rodríguez J, Puello P. Characterization Model of Software Architectures Recovery Process. Indian Journal of Science and Technology. 2018; 11(1):1–10. Crossref.
5. Eisenbarth T, Koschke R, Simon D. Locating features in source code. IEEE Transactions on Software -Engineering. 2003; 29(3):210–24. Crossref.
6. Favre JM. Cacophony: Metamodel-driven software architecture reconstruction. IEEE 11th Working Conference on Reverse Engineering. 2004 November; p. 204–13.
7. Callo Arias TB, Avgeriou P, America, P, Blom K, Bachynskyy S. A top-down strategy to reverse architecting execution views for a large and complex software-intensive system: An experience report. Science of Computer Programming. 2011; 76(12):1098–112. Crossref.
8. Boussaidi GE, Belle AB, Vaucher S, Mili H. Reconstructing architectural views from legacy systems. IEEE 19th Working Conference on Reverse Engineering. 2012; p. 345–54. Crossref.
9. Garcia J, Krka I, Medvidovic N, Douglas C. A framework for obtaining the ground-truth in architectural recovery. Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture. 2012; p. 292–96. Crossref.
10. Pinzger M, Gall H, Girard JF, Knodel J, Riva C, Pasman W,Wijnstra JG. Architecture recovery for product families. In Software Product-Family Engineering. Heidelberg; Springer, Berlin. 2003; p. 332–51.
11. Kang S, Lee S, Lee D. A framework for tool-based software architecture reconstruction. International -Journal of Software Engineering and Knowledge Engineering. 2009; 19(2):283–305. Crossref.
12. Stoermer C, Brien L, Verhoef C. Moving towards quality attribute driven software architecture reconstruction. Proceedings 10th Working Conference on Reverse Engineering (WCRE). 2003 November; p. 46–56.
13. Guo GY, Atlee JM, Kazman R. A software architecture reconstruction method. Springer US. 1999; p. 15–33. Crossref.
14. Deursen VA, Hofmeister C, Koschke R, Moonen L, Riva C. Symphony: View-driven software architecture reconstruction. Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA). 2004 June; p. 122–32. Crossref.
15. Vasconcelos A, Werner C. Evaluating reuse and program understanding in ArchMine architecture recovery approach. Information Sciences. 2011; 181(13):2761–86. Crossref.
16. Kazman R, O'Brien L, Verhoef C. Architecture reconstruction guidelines. Third Edition. Carnegie Mellon University. Software Engineering Institute, Pittsburgh. 2003; p. 1–43.
17. Kitchenham BA, Budgen D, Brereton OP. Using mapping studies as the basis for further research - A participant observer case study. Information and Software Technology. 2011; 53(6):638–51. Crossref.
18. Monroy M, Arciniegas JL, Rodríguez JC. Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura. Información tecnológica, 2016 September; 27(5)201–20
19. Yin RK. Case study research: Design and methods. Sage publications; 2013.
20. McGregor SL, Murnane JA. Paradigm, methodology and method: Intellectual integrity in consumer scholarship. International Journal of Consumer Studies. 2010; 34(4):419–27. Crossref.
21. Ducasse S, Pollet D. Software architecture reconstruction: A process-oriented taxonomy. IEEE Transactions on Software Engineering. 2009 April; 35(4):573–91. Crossref.