Indian Journal of Science and Technology, *Vol 11(1), DOI: 10.17485/ijst/2018/v11i1/118364, January 2018*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Characterization Model of Software Architectures Recovery Process

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

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

Abstract

Objectives: There are multiple approaches for processes to recover the architecture of a software product; however, a catalog has not been found, indicating when and how they should be used. The objective of this work is to propose a characterization model of the Software Architectures Recovery Process (SARP), which serves as a reference to establish the structure of the mentioned catalog. **Methods:** The model was obtained by applying the pattern matching technique to establish the common aspects of all the approaches identified in the literature review. **Findings:** The results of the evaluation of the model reveal its usefulness as a reference to establish the structure of the catalog, indicating when and how a specific process of architectures recovery should be used. **Applications:** The defined model is a new way of structuring and documenting SARP, which facilitates the understanding, analysis and selection of approaches to recover the architecture of a software product.

Keywords: Architecture Recovery, Characterization Model, Reverse Engineering

1. Introduction

To guarantee the evolution of a software product it is necessary to know its architecture. Unfortunately, architectural information, when available, it is often outdated and incorrect, or inappropriate for the task at hand.[1] Architecture recovery is a reverse engineering approach, whose main objective is to reconstruct the architectural views of a software product.[2] Previous work presents a broad list of existing approaches to recover architectures.[3-5]

These approaches describe how the SARP should be performed. In essence, they all follow the generic process formulated.[6] That includes the following main activities: Data extraction: Also known as collection, it consists in identifying the elements that make up the software product and the relationships between them, using appropriate mechanisms of extraction from the raw data of the system under study. Knowledge organization: In this activity the data obtained in the extraction stage are represented in a way that allows them to be stored and retrieved in an easy and efficient way, making possible the analysis of the artifacts and their relationships, it is also known as abstraction. Information Exploration: Provides the mechanisms for navigating, analyzing and presenting the results of the reverse engineering process. Navigation allows the architect to move through the information structures generated in the knowledge organization phase. To make the analysis possible, information that is not explicitly available is derived and extracted, generating views that may help the understanding of the system under study. The presentation of the results uses strategies that facilitate their understanding, making navigation and analysis possible.

However, each approach introduces elements and approaches that represent values for the process, depending on the situation where the need for architecture recovery is originated. For example, Symphony is a view-driven process that combines common patterns and good practices of reverse engineering to retrieve the architec-

*Author for correspondence

ture of a software product using views and architectural points view. It has two phases, the first one is the reconstruction of the design, where the problem is analyzed, the points of view that are intended to be reconstructed are selected, the views of the sources are defined, and the mapping rules of the sources are designated to the views to be achieved. This process allows a double result, the reconstruction of the design and the reconstruction of the architecture.

Some approaches are exposed in the same sense, such as mentioned in the article which is aimed at the reconstruction of software architecture driven by quality attributes that in addition to reconstructing the architecture, it makes possible their evolution; PuLSE-DSSA recovers the architecture of program families defines a process driven. On the other hand is oriented to the recovery concepts, He are based on data mining propose a process based on modeling of characteristics.[7-13]

The problem arises when selecting one of these approaches to recover the architecture of a software product, since there are more than a hundred and there is no catalog that indicates when and how they should be used in the presented article.[14,15] This implies that there is a high risk of selecting a approach that is not suitable for the circumstances of the context requires from the recovery of the architecture, which directly affects the relevance of the results obtained. The absence of unified criteria for describing architectural recovery processes makes it difficult to understand and apply existing approaches. An additional effort is required by studying and analyzing each of the existing approaches to select the most appropriate.

The main contribution of this work is the definition of a Characterization Model of SARP. It is aimed at software engineers who participate in SARP, or who define methodological approaches to recover architectures. The defined model serves as a reference to establish the structure of the catalog that indicates when and how to use a specific approach to recover the architecture, which contributes to the solution of the problem. In addition, the detailed characterization of 12 approaches is presented, whose were selected because include activities, techniques, or complementary instruments to the generic process defined.[6] In the next section the methods and tools used for the development of the research are presented. Then the results are explained presenting the characterization model. After, the model is applied to 12 architectures recovery approaches and the results are analyzed. Finally, the conclusions are presented.

2. Materials and Methods

The research was carried out in three phases. In the first phase, a review of the literature was made applying the methodological approach established.[16] The results of this phase were published in the second phase, the documentation identified in the previous phase was analyzed and the pattern matching technique was applied to establish the aspects common to all the approaches.[17] As a result of this phase the characterization model of the SARP was obtained.

In the third phase the characterization model was evaluated applying it to 12 approaches for architectures recovery. Only those approaches defining new activities, techniques or instruments that complement the generic process formulated were selected.[6] The analysis of the results was made taking into account each of the aspects defined in the characterization model. The characterization model is represented by the Unified Modeling Language (UML), through a domain model to facilitate the interpretation.

3. Results and Discussion

In this section, the characterization model of the SARP is described, explaining the formed elements and the existing relationships between them. Then the characterization of 12 approaches is presented applying the defined model. Finally the analysis of the results obtained is made and a future work is proposed.

3.1 Characterization Model of the SARP

The proposed characterization model is composed of 11 elements that are related to each other as shown in Figure 1. Each of these elements is explained below. The environment: it is the physical or situation context in which the SARP is considered. The environment of a system is determined by the set of circumstances that influence the system. It includes developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.[18] The SARP can be presented in the following environment.[19] Software production, Computer security, Forensic and Education computing. The environment determines the circumstances that define the approach and the purpose of the SARP.

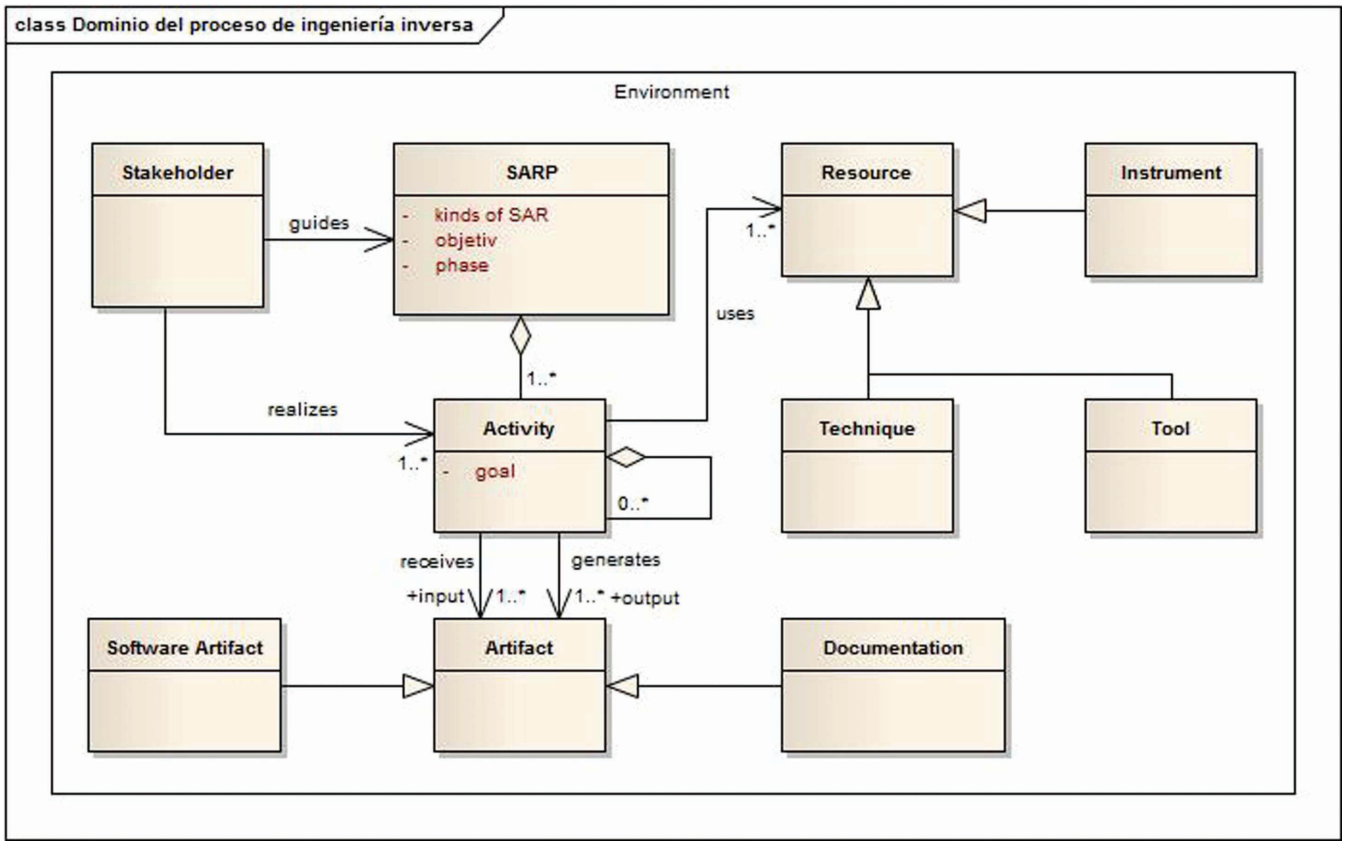

Figure 1. Conceptual model.

The SARP refers to the set of activities that are performed under a logical and temporal order organized in phases. A phase represents the states which the process goes through, depending on compliance with established milestones. All process is done in a specific environment, under a certain approach and establishes one or several objectives. The objective of the process establishes the purpose to be achieved and depends on the environment in which it is carried out. Some possible objectives that can be raised in an SARP are: the reconstruction of documentation, identification of assets for reuse, conformity analysis, analysis for the identification of patterns, aspects, characteristics, roles, collaborations, among others.

There are three kinds of SARP, depending on the referent used to start with it. The first one is "Bottom-Up", it starts with the low-level knowledge represented in the source code models, and continues making progressive processes of abstraction until reaching the level of architecture, following the approaches of Tilley et al also known as extraction, abstraction and presentation.

The second approach is "Top-Down", starting with high-level knowledge represented in requirements or architectural styles, which serve to formulate a hypothesis of the architecture, which is finally compared to the representation obtained directly from source code. The third type of approach combines the two above, therefore it is known as hybrid; because on the one hand, knowledge is abstracted using several techniques (Bottom-Up), while on the other hand, knowledge at the highest level is refined (Top-Down) and compared to the extracted one using the Bottom-Up approach.

The stakeholder is an individual, team, organization, or classes thereof, having an interest in a system. There are different types of stakeholders, which are determined by the environment in which the process is carried out[19]. The stakeholder is responsible for directing the process and carrying out the activities. An activity comprises a set of operations or tasks typical of a person or entity. Each activity establishes one or several goals that incrementally contribute to the achievement of the proposed objectives

for the SARP. An activity can be made up of several sub-activities or tasks organized in accordance with a work plan.

All activity receives input artifacts and generates output artifacts. An artifact is a piece of information used or produced in the software development process.[20] Artifacts may be represented by documents such as software models; or by architectural elements such as source files, databases, executable, etc. In addition, all activity requires resources to meet the established goals. The resources are represented by tools, techniques and architecture recovery tools. The instruments are elements of support that serve to guide or record the activities and the results obtained in each of them. An example of an instrument is the work plan. It establishes the logical and temporal order of the sequence of activities to be carried out, indicating the people responsible, the resources to be used and the results to be obtained.

A technique comprises the set of guidelines and tasks that must be performed to achieve an established goal for the activity. There are specific techniques for data extraction, organization of knowledge and exploration of information. Data extraction techniques are classified in techniques of static analysis and techniques of dynamic analysis.[2,4] The techniques of knowledge organization are classified in 1) manuals, 2) semi-automatic: Relational algebra, relational partition algebra, Reflection model, Prolog, Rigi graph transformation, SGL queries, ad-hoc query language; and 3) fully automatic: based on clustering: coupling, file names, concept analysis and type inference. Information exploration techniques are classified into navigation techniques, analyzing and presenting the results of the SARP.

Tools are the software systems that automate the performance of specific tasks of the process. There is an extensive list of tools for architecture recovery.[2,3] The literature review allowed identifying 115 specialized tools in reverse engineering, of which 34 recover aspects related to the design, 65 recover the architecture of the product, 5 recover the structure of the database and 11 are for visualization.[26] For a better understanding of reverse engineering tools and therefore, to facilitate their selection in an SARP, an Ontological Model was defined for contexts of use of reverse engineering tools.[21]

3.2 Application of the Model

The characterization model was applied to 12 SARP, selected from the results of the literature review. Only those approaches that included activities, techniques, or complementary instruments to the generic process defined by *Tilley et al.,* were selected. The result of the characterization is presented, represented in tables that describe each one of the characteristics defined in the model.

The title of each table corresponds to the name of the selected approach and has the respective bibliographic reference. The first two lines of the table refer to the properties of the SARP: The objective and the kind of SARP. Then the artifacts used by the process as input and output artifacts are detailed. Subsequently, the activities included in each approach are listed, together with their respective techniques, tools and instruments. The information was taken directly from the publications made by the authors on the approach. The empty fields indicate that no information about the feature was found.

3.3 Analysis of Results

The characterization model was obtained from the abstraction of the characteristics described in the approaches identified in the literature review. It complements the work done because it integrates them and includes elements such as environment and stakeholders, In addition, it makes an abstraction of the different types of resources that are used in the SARP, and the artifacts that are used and generated during the process. Therefore, the model can be used as a reference to establish the structure of the catalog that indicates when and how a specific SARP should be used. This facilitates the understanding, analysis and selection of approaches to recover the architecture of a software product in different process as software maintenance, reverse engineering, re-engineering and others context.[22-28]

The model defines a structure that serves to document SARP, as shown in Tables 1-12. It also serves as an instrument to support the tasks of evaluation and selection of SARP, since it facilitates their understanding and comparison. The model is limited to establish the characteristics of the generic process of SARP. Specific aspects of the elements that make up the model are not detailed to facilitate their understanding and use. In addition, the detailed description of these aspects can be found in the literature. For example, the environment and stakeholders are described in depth in OMG describes the detailed information on the techniques; and et al characterize the tools.

Table 1. Locating features in source code

Objective:	Re-documentation		
Kind of SAR:	Hybrid		
Input artifacts:	Source code, expert knowledge		
Output artifacts:	Graphs		
Activities	**Techniques**	**Tools**	**Instruments**
Scenario creation	Manual inspection		
Static dependency graph extraction		Bauhaus, Rigi	
Dynamic analysis	Profiling		
Interpretation of concept Lattice	Concept analysis, Scenario Feature Mapping	Graph Viz	
Static dependency analysis	Concept Lattice		
Incremental analysis			

Table 2. ARM (Architecture Reconstruction Method)

Objective:	Conformance		
Kind of SAR:	Bottom-up		
Input artifacts:	Source code		
Output artifacts:	Recognized pattern instances		
Activities	**Techniques**	**Tools**	**Instruments**
Developing a concrete pattern recognition plan	Pattern rules		
Extracting a source model	Relational algebra	SAAMTool, IAPR, LSME, Imagix, SNiFF+	
Detecting and evaluating pattern instances		Dali	
Reconstructing and Analyzing the architecture	Visualization	Rigi	

Table 3. Symphony

Objective:	Re-documentation/Conformance/Evolution		
Kind of SAR:	Bottom-up		
Input artifacts:	Source code		
Output artifacts:	Architectural views		
Activities	**Techniques**	**Tools**	**Instruments**
Problem Elicitation	Structured workshops, checklists, role playing, and scenario analysis		Stakeholder/ View tables
Concept Determination	Mapping		
Data Gathering	Manual Inspection, Lexical Analysis, Syntactic Analysis, Fuzzy parsing, Island Grammars, Semantical Analysis	grep y perl scripting	Techniques classification
Knowledge Inference	Manual, automatic, or semi-automatic	SHRiMP, Rigi, PBS, and Bauhaus	Techniques classification
Information Interpretation	Presentation (visualization) and interaction		

Table 4. Architecture reconstruction guidelines

Objective:	Re-documentation/Conformance/Analysis/Evolution		
Kind of SAR:	Bottom-up		
Input artifacts:	Source code		
Output artifacts:	Subsystems		
Activities	**Techniques**	**Tools**	**Instruments**
Information Extraction	Lexical analysis, Syntactical analysis, code instrumentation, profiling		Guiding Principles for Choosing Types of Extraction
Database Construction		ARMIN	Guidelines for database construction
View Fusion		ARMIN	Guidelines for view fusion
Architectural View Composition	Visualization		Guidelines for visualization
Architecture analysis	ATAM		

Table 5. PuLSE-DSSA

Objective:	Evolution		
Kind of SAR:	Hybrid		
Input artifacts:	Source code /Documentation / expert knowledge		
Output artifacts:	Hierarchical graphs, UML logical diagrams, Message sequence charts, HTML reports		
Activities	**Techniques**	**Tools**	**Instruments**
Determine Architectural Views and Concepts			
Architecture Recovery (Extraction, abstraction, visualization)	Semi-automatic Component Recovery, Pattern-Supported Architecture Recovery		
Analysis of Recovered Architecture			Guidelines for Architecture Comparison Process
Design of a Reference Architecture			Guidelines for process of the reference architecture

Table 6. QADSAR (Quality Attribute Driven Software Architecture Reconstruction)

Objective:	Analysis		
Kind of SAR:	Hybrid		
Input artifacts:	Quality attributes scenarios / Source code		
Output artifacts:	Architectural views		
Activities	**Techniques**	**Tools**	**Instruments**
Scope Identification	Interview the expert, Conduct a reconstruction workshop, document analysis		
Source Model Extraction	Lexical analysis, profiling, instrumentation	ARMIN	
Source Model Abstraction	Relation Partition Algebra, Tarski Algebra, aggregate functions, pattern matching		
Element and Property Instantiation		TimeWiz	
Quality Attribute Evaluation.	Quality attributes scenarios		

Table 7. Cacophony

Objective:	Re-documentation		
Kind of SAR:	Hybrid		
Input artifacts:	Source code		
Output artifacts:	Architectural views		
Activities	**Techniques**	**Tools**	**Instruments**
Metaware domain and asset analysis		GSEE	Inventory of metaware items, Metamodels integration
Metaware requirement analysis			
Metaware specification			Metamodel-driven visualization
Metaware implementation			
Metaware execution			
Metaware evolution			

Table 8. Reconstructing Architectural Views from Legacy Systems

Objective:	Re-documentation		
Kind of SAR:	Bottom-up		
Input artifacts:	Source code		
Output artifacts:	Dependency graphs		
Activities	**Techniques**	**Tools**	**Instruments**
Extract KDM representation			
Select KDM entities and relations that are relevant for an architectural view			
Select a decomposition algorithm and set necessary parameters			
Refine the resulting components view			
Document the resulting components			
Build views using clustering algorithms	Clustering	IRIS™	

Table 9. ArchMine

Objective:	Conformance / Reutilization		
Kind of SAR:	Bottom-up		
Input artifacts:	Source Code		
Output artifacts:	Sequence diagrams an package diagrams in UML		
Activities	**Techniques**	**Tools**	**Instruments**
Static structure extraction and use case scenarios definition		Odyssey	Guidelines form definitions use case scenarios
Dynamic analysis		TraceMining tool	
Architectural elements reconstruction	Data mining	ArchMine	Concept mapping for mining association rules
Dynamic view reconstruction			
Visualization		ArchMine	

Table 10. FASAR: The Framework for Automating Software Architecture

Objective:	Evolution		
Kind of SAR	Top-Down		
Input artifacts:	Source code and expert knowledge		
Output artifacts:	Architectural views		
Activities	**Techniques**	**Tools**	**Instruments**
Preparation Phase:	Manual inspection		
Problem Definition			
Hypothetical Views Definition			
Reconstruction Phase	Mapping		List of software architecture tools
Static Views Reconstruction			
Dynamic Views Reconstruction			

Table 11. A top-down approach to construct execution views of a large software-intensive system

Objective:	Re-documentation		
Kind of SAR:	Top-Down		
Input artifacts:	Documentation, execution system		
Output artifacts:	Execution profile, execution concurrency and resource usage		
Activities	**Techniques**	**Tools**	**Instruments**
Reconstruction design			
Reconstruction execution:			Execution meta-model
Task identification	Profiling / Instrumentation	Python scripts	Execution views
Interpretation of runtime information	Mapping rules	Python scripts	
Construction of execution model		Python scripts y Graphviz	

Table 12. A Framework for Obtaining the Ground-Truth in Architectural Recovery

Objective:	Re-documentation		
Kind of SAR:	Hybrid		
Input artifacts:	Documentations, domain literature, system context		
Output artifacts:	Utility components		
Activities	**Techniques**	**Tools**	**Instruments**
Gather domain and application information	Manual inspection		Mapping principles
Select a generic recovery technique			
Extract relevant implementation-level information		IBM RSA y Class Dependency Analyzer	
Apply generic recovery technique	Existing recovery techniques		
Utilize domain and application principles			
Identify utility components			
Pass authoritative recovery to certifier			
Modify groupings according to certifier's suggestions			

On the other hand, the literature review revealed that all the approaches identified are located in the context of software production. This demonstrates the need to define approaches for the recovery of architectures that fit the interests of participants in contexts such as computer security, computer forensics and education. With respect to the approaches characterized, it can be assumed that t the majority has as objective the re-documentation, and uses as input artifact the source code.

It is also observed that in essence all the approaches maintain the basic structure of the process defined by Tilley et al. Some of them complement the process including a previous activity to understand the problem, establish the scope, define the objective and design the work plan. Likewise, in addition to recovering the views of the software product, it is evident that there is a need to have resources that facilitate the interpretation and analysis of the results obtained in the data extraction phase, in order to achieve more relevant results to the context and the situation in which the recovery process is performed.

Additionally, the analysis indicates a tendency to use tools developed by the same authors, a circumstance that justifies the large number of existing tools and the complexity when selecting the most convenient one. Likewise, the results reveal that there are many activities for which there are no defined techniques, tools of support, nor instruments, which makes it difficult to understand and carry out the process.

Finally, the absence of unified criteria is observed to describe the artifacts that are obtained as a result of the SARP. Some authors generally do it, indicating architectural views, subsystems, and components; while others do so in a specific way, indicating the type of representation and even the language they use to express it. This also makes it difficult to understand the approaches and the possibility of comparing them when selecting one of them.

4. Conclusion

A characterization model was defined for the SARP. The model was obtained by applying the pattern matching technique to the approaches identified in the literature review. The model was applied to 12 approaches and concluding: The model serves as a reference to establish the structure of the catalog that indicates when and how to use a specific SARP. The model is useful for structuring, documenting and evaluating the approaches that define SARP. It is necessary to define approaches for the architectures recovery that fit the interests of the participants in contexts such as computer security, computer forensics and as well as education. It is also necessary to establish unified guidelines to describe the artifacts that are obtained as a result of the SARP.

5. References

1. Deursen VA, Hofmeister C, Koschke R, Moonen L, Riva C. Symphony: View-driven software architecture reconstruction. In Software Architecture Proceedings Fourth Working IEEE/IFIP Conference on IEEE; 2004 Jun. p. 122–32.
2. Ducasse S, Pollet D. Software architecture reconstruction: A process-oriented taxonomy. IEEE Transactions on Software Engineering. 2009 Apr; 35(4):573–91. Crossref.
3. Monroy M, Arciniegas JL, Rodríguez JC. Software Architecture Recovery: A Systematic Mapping Study. Informacion tecnologica. 2016 Sep: 27(5):201–20.
4. Monroy M, Arciniegas JL, Rodríguez JC. A Methodological Approach for Characterization of Reverse Engineering Methods. Informacion tecnologica. 2013 Sep; 24 (5):23–30.
5. Monroy M, Arciniegas JL, Rodríguez JC. Characterization of Reverse Engineering Tools. Información tecnológica. 2012 Nov; 23(6):31–42.
6. Tilley SR, Smith DB, Paul S. Towards a Framework for Program Understanding. In: Program Comprehension Fourth Workshop; 1996 Mar. p. 19–28. Crossref.
7. Stoermer C, Brien L, Verhoef C. Moving towards quality attribute driven software architecture reconstruction. Proceedings 10th Working Conference on Reverse Engineering (WCRE); 2003 Nov. p. 46–57.
8. Medvidovic N, Jakobac V. Using Software Evolution to Focus Architectural Recovery. Automated Software. 2006; 13(2):225–56. Crossref.
9. Pinzger M, Gall H, Girard JF, Knodel J, Riva C, Pasman W, Wijnstra JG. Architecture recovery for product families. In Software Product-Family Engineering, Springer, Berlin, Heidelberg; 2003. p. 332–51.
10. Favre JM. Cacophony: Metamodel-driven software architecture reconstruction. In 11th Working Conference on Reverse Engineering IEEE; 2004 Nov. p. 204–13. Crossref.
11. Riva C, Brien OL, Verhoef C. Architecture reconstruction in practice. In Software Architecture, Springer US; 2003 Nov. p. 159–73.
12. Vasconcelos A, Werner C. Evaluating reuse and program understanding in ArchMine architecture recovery approach. Information Sciences. 2011; 181(13):2761–86. Crossref.

13. Pashov I, Riebisch M. Using feature modeling for program comprehension and software architecture recovery. In Engineering of Computer-Based Systems. Proceedings 11th IEEE International Conference and Workshop on the IEEE; 2004 May. p. 406–17. Crossref.
14. Boussaidi G, Belle E, Vaucher AB, Mili H. Reconstructing architectural views from legacy systems. In Reverse Engineering (WCRE) 19th Working Conference on IEEE; 2012 Oct. p. 345–54. Crossref.
15. Koschke R. Architecture reconstruction: Tutorial on reverse engineering to the architectural level. Lecture notes in computer science; 2009. Crossref.
16. 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.
17. Yin RK. Case study research: Design and methods. Sage publications; 2013.
18. Farenhorst JF, Lago R. Standard I: ISO/IEC 42010 Systems and Software Engineering Recommended Practice for Architectural Description of SoftwareIntensive Systems. ISOIEC 42010; 2011.
19. Monroy M, Arciniegas JL, Rodríguez JC. Characterization of the contexts of use of reverse engineering. Información tecnológica. 2017 Jul; 28(4):75–84.
20. OMG Meta Object Facility (MOF) Core Specification versión 2.5. http://www.omg.org/spec/MOF/2.5/. Date accessed: 06/2015.
21. Monroy M, Arciniegas JL, Rodríguez JC. Ontological Model for Contexts of use of Reverse Engineering Tools. Información tecnológica. 2017; 27(4):165–74.
22. Eisenbarth T, Koschke R, Simon D. Locating features in source code. IEEE Transactions on Software Engineering. 2003; 29(3):210–24. Crossref.
23. Guo GY, Atlee JM, Kazman R. A software architecture reconstruction method. Springer US; 1999. p. 15–33. Crossref.
24. Kazman R, O'Brien L, Verhoef C. Architecture reconstruction guidelines Carnegie Mellon University. Third Edition. Software Engineering Institute, Pittsburgh; 2003.
25. 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.
26. 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.
27. Garcia J, Krka I, Medvidovic N, Douglas C. A framework for obtaining the ground-truth in architectural recovery. Paper presented at the Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA; 2012. p. 292–6. Crossref.
28. Kalia A, Sood S. Concerns in Maintaining Reusable Software Components and the Possible Solutions. Indian Journal of Science and Technology. 2017; 10(23):1–4. Crossref.