ARCo: Architecture Recovery in Context

Martin Monroy
Systems Engineering Program
Universidad de Cartagena, Cartagena, Bolívar, Colombia
Email- mmonroyr@unicartagena.edu.co

Martin Pinzger
Software Engineering Research Group
Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
Email- Martin.Pinzger@aau.at

Jose Luis Arciniegas
Telematics Department
Universidad del Cauca, Popayán, Cauca, Colombia
Email- jlarci@unicauca.edu.co

Abstract- Software architecture recovery is an activity which is carried out in contexts such as software production, forensic computing, computer security and education. Each context is determined by an ambit, a set of resources and situations that affect a group of stakeholders. There are multiple techniques, methods and frameworks to recover the architecture of a software product, however, they assume that architecture recovery is exclusive of the software production context. This assumption makes its application difficult in other contexts, because the differences between ambits, resources, situations or interests of the stakeholders of the process are not taken into account. We introduce ARCo, a framework for the recovery and analysis of architectural views, shaped by a conceptual descriptive system and a set of instrumental elements of operational type. To define ARCo, we analyze the approaches of architecture recovery using the pattern matching technique. We evaluate ARCo through a single case study in different contexts; one was the context of education to support the teaching and learning process, and the other, the context of software development to recovery the documentation of a software product. ARCo is made up of a conceptual model, some guidelines, an architecture recovery methodology and a query mechanism. ARCo allowed to recover the architecture of software products in different contexts, attending to the particular needs of each one of them. We conclude that ARCo offer a new approach for software architecture recovery, since it involves the context where the process is carried on, and hide its complexity from stakeholders.

Keywords – ***Architecture recovery, context, reverse engineering, architectural views, ARCo***

I. INTRODUCTION

There is a large number of approaches to recover the architecture of software products, which are classified by methods, techniques, tools and frameworks [1]. Methods are focused on the process and define the logical order of the activities required to recover the architecture of a software product. Techniques detail the way every activity that comprises the process should be fulfilled. Tools implement techniques to automate the process and frameworks comprise methods, techniques and tools. The approaches identified apply the activities specified by Tilley et al. [2]: data extraction, knowledge organization and information exploration. Some approaches focus on the analysis of software products [3][4], others on supporting the evolution [5][6], on reconstructing the architecture of the system [7][8][9][10] or on the re-documentation of the product [10][11][12]. All these approaches are particularly oriented to the process of software engineering, in activities such as maintenance, software verification, re-documentation and asset reuse [1][13]. On the other hand, the recovery of architectures as a reverse engineering approach, is an activity performed also in other contexts such as education [14][15][16][17][18], computer security [19][20][21][22] and computer forensics [23], besides the software production.

Each context is determined by a scope, situations, resources, purposes and stakeholders, who have concerns and particular characteristics [13]. In this scenario, the following questions arise when architecture recovery in contexts other than software production is performed: How to guide the architecture recovery process taking into account the characteristics of the stakeholders and their concerns?, How to guide the recovery process of architectures taking into account the situations present in the context where the process is carried out?, How to guide the recovery process of architectures taking into account the scope and purpose of the context where it is performed?, and How

to hide the complexity of the process of recovering architectural views for those stakeholders who are not experts in reverse engineering?

The recovery of architectures is a field of research which has attracted the attention of researchers over the past recent years from multiple perspectives [7][13][24]. Its main objective is to reconstruct the architectural views of a software product [25]. According to the ISO/IEC/ IEEE 42010: 2011 standard, architecture comprises the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [26]. An architectural view is a work product expressing the architecture of a system from the perspective of specific system concerns, for which a type of model as a way of representation is used (graphs, diagrams, Petri networks, etc.). There are several approaches which define the points of view and the views that describe the architecture of a software system [26][27][28][29], which forces to select an approach to represent an architecture, which depends on the context and the situation [29]. This circumstance increases the complexity of the architecture recovery process, because it demands for each approach to take into account situations, resources available, stakeholders along with their limitations and purposes.

In this work, a framework for the recovery and analysis of software architectures is proposed, based on the circumstances related to the availability of resources, the stakeholders involved, their concerns and the situations that arise at the moment of recovering the architecture of software products. As a result, ARCo (Architecture Recovery in Context) was obtained, a framework composed by a conceptual system and an instrumental part. The conceptual system is represented by a conceptual basis, a theoretical basis and a conceptual model. The instrumental part includes a methodology for the recovery of software architectures, a query mechanism, a set of guidelines for the use of ARCo and the characterization of the contexts of use of reverse engineering. To evaluate ARCo, a single case study was conducted in two different contexts, namely, first in an educational and second, in a software development context. Validity of construction, external validity and reliability were evaluated to judge the design quality of the case study [30].

The main contributions of this work are: 1) A conceptual model to recover and analyze architectural views, based on ISO/IEC/IEEE 42010, ISO/IEC 19506 and UML standards. 2) A methodology for the recovery and analysis of architectural views, depending on the context where the need arises, which allows users to obtain more detailed and accurate results compared to other approaches. 3) The definition and characterization of the contexts of use of reverse engineering. 4) The inclusion of context analysis in the architecture recovery process, represented by the scope, situations, resources, stakeholders and their concerns. 5) The design of the query mechanism to support the analysis of architectural views and its implementation in the QModel-XMI prototype. 6) The design, results and the database of the case study, with its two units of analysis, which can be used as a guide for the application of ARCo.

It is highlighted that ARCo does not substitute the existing approaches, but it complements them by integrating the aspects related to the scope, resources, stakeholders along with their purposes and the situations that arise in the different contexts where the architecture recovery process of a software product is performed into its processes. We argue, that this integration makes architecture recover more feasible and accurate for those users who are interested but not experts in reverse engineering, as demonstrated by the case study.

The rest of the paper is organized as follows: Section II presents the related works. Next, Section III, introduces the proposed framework. Section IV explains the framework's conceptual system and Section V describes the framework's instrumental part. Section VI presents the evaluation of the proposed framework in educational and Software development contexts. Section VII analyzes and discusses the results of the work, and finally, Section VIII presents the conclusions.

II. RELATED WORKS

The study of software architecture recovery has focused on the definition of techniques that specialize in reconstructing the system documentation [31][32] [33][34], in the compliance review [22][35], in the evolution [24][36][37] and analysis [3][38] of the product, or in achieving one or more of these purposes at the same time [39][40]. These approaches are especially focused on the specific activities of architecture recovery, than on the process itself. Multiple tools have been developed to automate architecture recovery [41][42][43]. These tools facilitate the process and make it more efficient for users. However, to leverage the tools, the procedure to recover the architecture of a software product must be clearly defined.

This procedure comprises the activities of the canonical process defined by Tilley et al. [2], which have been refined by several approaches. For instance, Symphony [44] focuses on general aspects of the architecture recovery process and on how to select the views to be reconstructed. Cacophony [32] defines a generic process driven by meta-models. Furthermore, Pinzger et al. define a process for architecture recovery for product families [45]; REM

[7] and Architecture reconstruction in practice [46] establish a general method for architecture recovery; QADSAR [3] focuses on the recovery of architecture from the perspective of quality attributes; Boussaidi et al. [8] define a framework based on KDM; Sartipi et al. [9] establish a process to extract three views of the system; and Garcia et al. [12] propose a framework that includes a set of principles and processes for recovering a system's ground-truth architecture. Each of these approaches defines a set of activities and establishes the logical order under which they must be performed to recover the architecture of a system. Boussaidi et al. [8] focus on the modernization of the product, and Sartipi et al. [9] on maintenance. Stoermer et al. [3] declare as contexts of architecture recovery: "improving the understanding of the software architecture of existing systems, improving the architecture itself, assessing quality attribute characteristics of these systems, improving the documentation of the architecture of these systems". All of these activities are typical of the software construction process.

For Favre [32] and Garcia et al. [12] the context refers to the physical environment or situation of the application, while for Tamburri and Kazman [7] it corresponds to the circumstances of product development (industrial and open-source). Only Van Deursen et al. [44] contemplate aspects related to the availability of resources and the interests of the stakeholders when recovering the architecture of the software product. However, this work does not explicitly take into account the situations, the characteristics of the stakeholders, the scope and the purpose of the context where the architecture recovery process takes place. This creates a breach that hinders the recovery of software architectures in contexts such as education, forensic computing and computer security. Therefore, it is necessary to define a framework to guide the recovery process and the analysis of architectures, considering the particular characteristics of the situation that surrounds the context where such process takes place.

III. ARCo FRAMEWORK

In this section, we introduce ARCo (Architecture Recovery in Context). ARCo consists of a conceptual basis, a theoretical basis and a conceptual model, which are derived from theoretical assumptions about the nature of the architecture recovery process and how to approach it. Figure 1 provides an overview of ARCo.

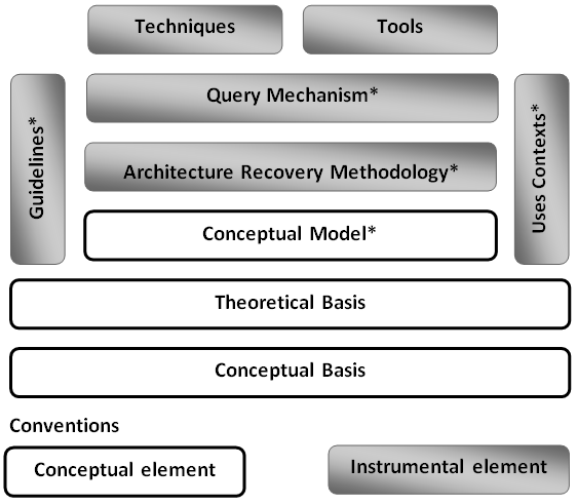

Figure 1. ARCo: Overview

The instrumental part of the framework, represented with gray color in Figure 1, comprises: 1) a set of guidelines, which determine the directives that guide the definition of the framework and how it should be used. 2) A methodology that uses reverse engineering techniques and tools, and defines how to complete the recovery and analysis process following guidelines, to meet the purposes of the specific context in which the process is performed. 3) A query mechanism to support the process of analyzing the recovered architectures. 4) The contexts of use in which it is necessary to recover the architectural views of software products. To differentiate our contributions, the name of the ARCo's elements that extend the state of the art, appears with an asterisk (*) in Figure 1.

IV. ARCo'S CONCEPTUAL SYSTEM

A conceptual model is an external, accurate, complete and consistent scientifically shared knowledge representation, which facilitates the understanding or teaching of systems or states of things in the world [47]. ARCo'S conceptual system is made up of the conceptual basis, the theoretical basis and the conceptual model. The conceptual basis and the theoretical basis are represented in the available literature about recovery and analysis of

architectural views. The proposed conceptual model provides a holistic view of reverse engineering and a detailed vision of the recovery and analysis of architectural views, facilitating the study and understanding of this field of knowledge.

The conceptual model is presented in two dimensions, the first corresponds to the holistic view of reverse engineering (Figure 2) and the second focuses specifically on the recovery and analysis of architectural views (Figure 3). Reverse engineering includes any method aimed at recovering the knowledge of a software system, to support the execution of activities which require the understanding of such system. This implies the existence of two relevant aspects in reverse engineering: The context in which the need to recover knowledge is presented, and the process that is performed to achieve the recovery of knowledge. That is why the conceptual model comprises these two elements, as shown in Figure 2.

The context is determined by the ambit, the resources, the situations that arise, the stakeholders and their purposes related to the recovery of architectural views, as it is explained in detail in [13]. On the other hand, the process of knowledge recovery is determined by a methodology that uses techniques, instruments and tools, to fulfill the purposes established by performing a set of activities that generate and require artifacts, according to the work plan.

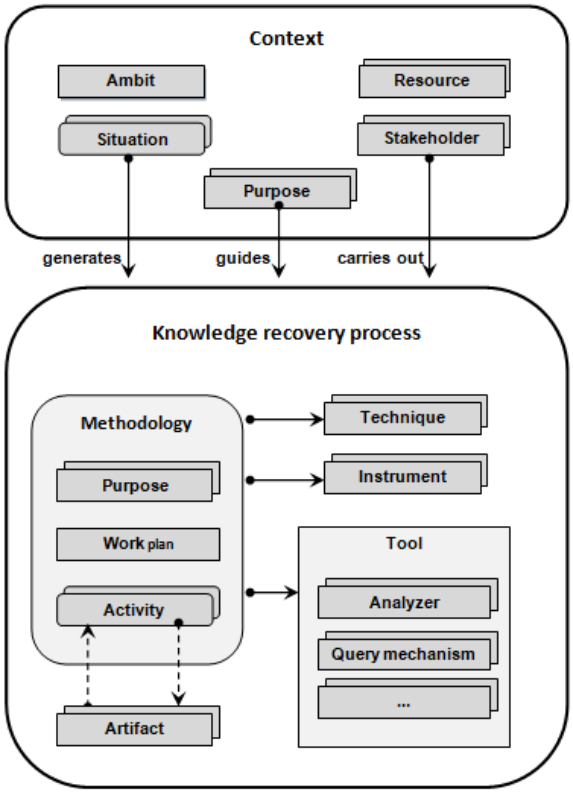

Figure 2. Holistic view

The detailed vision of the conceptual model focuses its attention on the understanding of each of the activities that tackles the generic architecture recovery process [2] and the artifacts that are required or generated in the development of these activities, as observed in Figure 3. To recover the implicit knowledge in the artifacts that compose the software product, these artifacts must be identified and decomposed into the parts that make them up, using data extraction techniques. To guarantee interoperability between reverse engineering tools, the KDM (Knowledge Discovery Metamodel) [48] specification is used to represent the software product under study, its elements, associations and operating environments. The elements of the source package and the KDM code package can be extracted by setting mapping rules from the programming language (or languages) of the product under study, because they are explicit in the syntax. However, the elements of the upper layers (abstraction and resources at runtime) are implicit, since they correspond to higher levels of abstraction. That is why they require incremental analysis starting from the primitive KDM representations.

Based on the KDM representation of the software product under study, the identified elements can be organized forming system models, applying knowledge organization techniques. Following the interoperability criteria, the generated models must be represented through mechanisms that make possible their interpretation, such as the use of the XML markup language. UML is used in the proposed conceptual model to guarantee this characteristic, representing the models obtained in the knowledge organization process by means of diagrams expressed in XMI. The views of the system can be composed of one or several models and be visualized by any tool that interprets the UML models in XMI. The views obtained depend on the objectives established when applying the methodology.

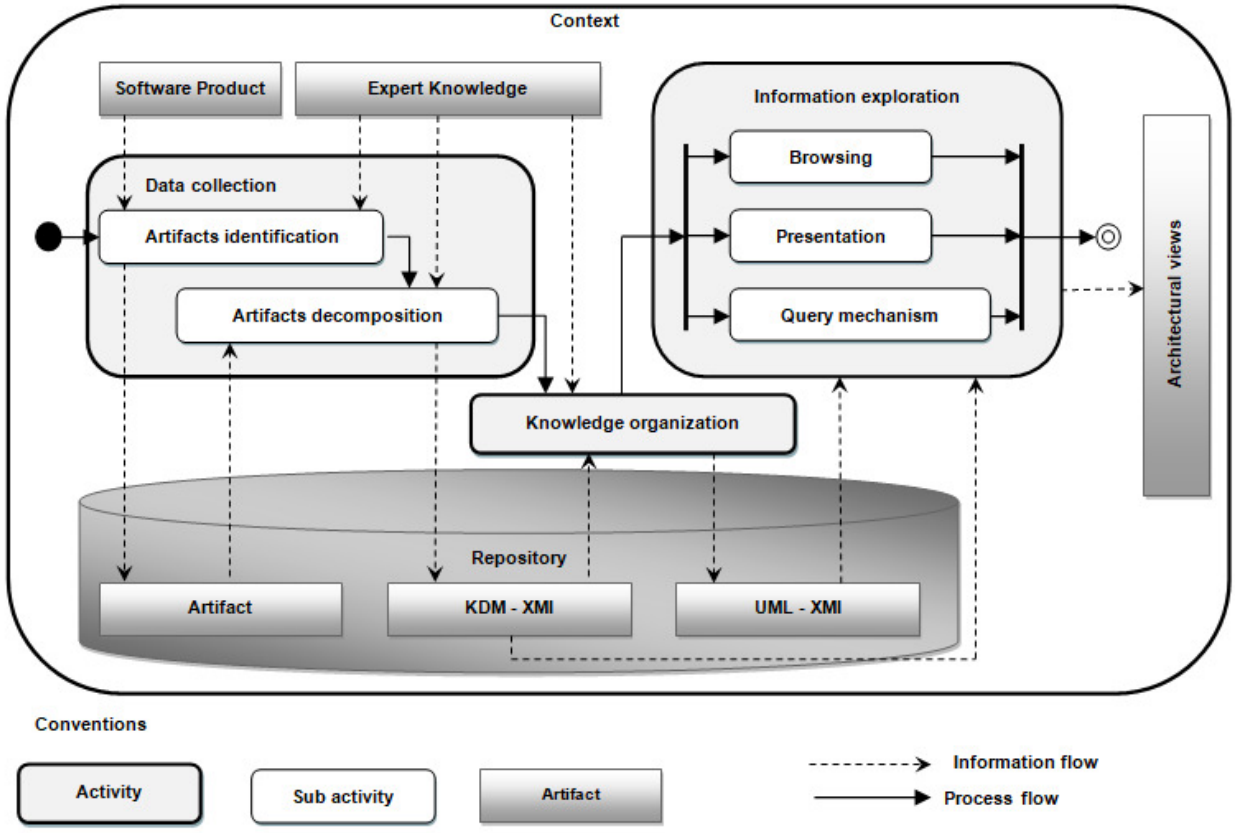

Figure 3. . Conceptual model

Every activity has an aim or intention; it receives inputs, applies techniques and generates outputs. The extraction of data is the first activity of the generic process of architecture recovery [2]. Its main purpose is to collect the data required to achieve the architectural views that are intended to be recovered. To achieve this purpose, the raw data of the software product under study and the knowledge of the expert are required as an input. This activity includes two tasks: the first consists of identifying the artifacts that constitute the software product, where manual inspection techniques are applied. As results of this task, repositories containing the software artifacts are obtained, such as executable files, binary files, source code files, configuration files, resource description, documentation images, among others; taking into account that the availability depends on the context in which the need to recover the architectural views arises. The second task is to decompose the identified artifacts to achieve a KDM representation of the system that is kept within an XMI repository. This is achieved by applying techniques such as lexical analysis, syntactic analysis, fuzzy analysis, grammatical islands, syntactic coincidences, data flow analysis, code instrumentation and profiling.

The second activity is to organize knowledge. It aims at representing the data obtained from the previous activity in such a way that storage and retrieval are easily and efficiently done, to allow an analysis of the artifacts and their relationships. As a result of this activity, models are obtained in UML, under an XMI representation, of the architectural views to be recovered. To achieve this, some techniques can be applied, such as: Relational algebra,

propositional algebra, Tarski algebra, reflection model, SQL queries, ad-hoc query languages, and grouping algorithms.

The last activity is information exploration. Its intention is to provide the mechanisms necessary to navigate, analyze and present the results of the recovery process. The presentation and navigation are achieved by the use of tools which allow the visualization of UML models in their XMI representation. To make the analysis possible, a query mechanism is defined, which derives and extracts information that is not explicitly available in the models retrieved, generating results that help understanding of the system under study.

V. ARCo's INSTRUMENTAL PART

In ARCo, the instrumental part is represented by the guidelines of the framework, the methodology for the recovery and analysis of architectural views, the query mechanism and the definition of the contexts of use.

5.1 Guidelines

The following guidelines were taken into account when defining ARCo:

Objective: to support the work of recovery and analysis of architectural views performed by those individuals involved in this activity, considering the context in which the situation under study is presented, so that the results are more relevant, efficient and precise; and assuming that not all of those interested are experts in reverse engineering.

Ambit: it includes the recovery and analysis of architectural views of object-oriented software products, independently of the technologies used for their development and deployment. It differs from the approaches identified in the review of the literature, because it expands the contexts of use of reverse engineering to fields of knowledge such as education, computer security and forensic computing, without being limited exclusively to them.

Standards use: the framework was developed based on: ISO / IEC / IEEE 42012: 2011 for the description of the architecture, ISO / IEC 19506: 2012 as a meta-model for the discovery of knowledge; and the specifications: OMG UML 2.5: 2015, OMG XML Metadata Interchange 2.5.1: 2015.

Integration: the conceptual model and methodology were defined as a result of the integration of the approaches identified in the literature review, assuming as central steps of the process those established in [2], because in essence all the approaches incorporate them.

The factors that influence the architecture recovery have also been taken into account [49], which can be classified as: 1) technological factors represented by the use of architectural patterns, programming paradigms, specific technologies, etc.; 2) factors of human nature, manifested in the skills, experience and knowledge about the domain of the problem and reverse engineering; 3) the availability of resources as reverse engineering tools, support instruments for the process, documents, standards, among others; and 4) contexts of use [13]. To achieve an efficient use of ARCo it is recommended: 1) To know the conceptual basis and the theoretical basis. 2) Understand the conceptual model proposed. 3) To know the techniques, tools and instruments. 4) To apply the methodology using the techniques and instruments proposed for each activity. 5) To count on experts in the domain of the problem and the application.

5.2 Methodology for the recovery and analysis of architectural views

Our methodology conforms to the four axioms of McGregor and Murnane [50]. The epistemological axiom is represented by the conceptual basis, the knowledge basis and the conceptual model of ARCo. The ontological axiom establishes as an object of study the recovery of architectural views under a specific context [13]. The axiological axiom defines the following methodological guidelines: 1) The proposed methodology corresponds to a generic process to support the recovery of architectural views. 2) The existing approaches are integrated so the results are relevant to the context and the situation where the problem arises. 3) The recovery process complies with the characterization model established in [51]. 4) The process defines the proposed goals, the activities that comprise it, the logical sequence of its execution, the input and output resources, as well as the techniques applied and the instruments that are used to achieve the proposed goals. Lastly, the logical axiom is constituted by the sequence of actions that make the recovery of architectural views possible.

In addition to the axioms, as depicted in Figure 4, the methodology establishes a process which sets objectives and it is performed under the definition of a work plan, organized in phases, where each one of them comprises a set of activities, meeting goals oriented to the achievement of the established milestones, to achieve the fulfillment of the formulated objectives for the process, relying on techniques and the use of diverse instruments and tools.

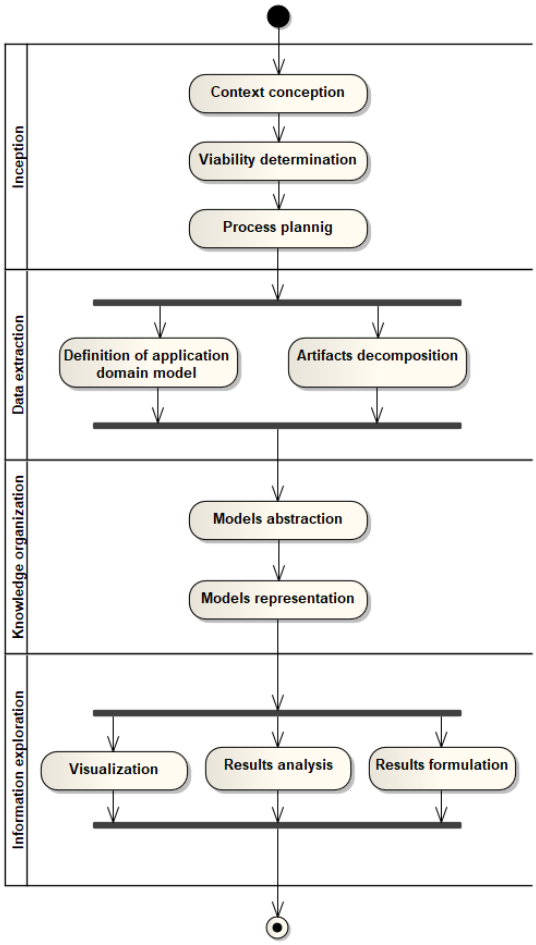

Figure 4. Overview of the methodology for architecture recovery and analysis

The process is structured in four phases. Each one of them establishes one or several milestones and a set of activities that are executed in a logical sequence, as shown in Figure 4. Every activity defines the goals to be achieved, the techniques required and instruments for its fulfillment. Similarly, for each activity the arguments are identified that justify it (motivation), the stakeholders, the artifacts that are required as input and the artifacts that are generated as output or result. If the activity leads to the completion of tasks, the aforementioned aspects are specified for each task. The order of execution of the activities that comprise each of the phases is established in the work plan, depending on the analysis of the context and the situation generated by the architecture recovery process.

Inception Phase: it aims to understand the context of the problem, identifying the ambit, the stakeholders and their interests, the available resources, the situation that generates the need to recover the architecture and the purposes to be achieved, that is why it contemplates three milestones that are achieved sequentially. The first is to specify the problem ambit, the second to define the feasibility of performing the process of architecture recovery and the third is to establish the work plan. To achieve these milestones, the following activities are executed:

1) *Context conception*: it consists of determining the ambit of the situation that generates the need, so two goals are established, the first is to identify the context and the second is to formulate the problem. The first goal is to define the context by describing its ambit, purposes, resources and stakeholders. The second goal involves the description of the situation that generates the need, pointing its causes, the circumstances of the context and the possible consequences that imply the problem, to perform the respective analysis and to establish the objective of the process of architecture recovery. It generates as output the document of problem description, which describes the context, all its aspects, the situation generating the need to recover the architecture, and the objective of the process is posed, achieving the first milestone of the inception phase: to specify the ambit of the problem.

2) Viability determination: It consists of determining if all the necessary resources to achieve the objective stated in the description of the problem are present. In this activity, a single goal is established: to identify if the project is viable or not. To achieve it, it is recommended to execute the following steps: 1) To define the views to be

recovered, considering the objective stated at the description of the problem. 2) To identify the software artifacts that are required to retrieve each view. 3) To verify the availability of the required software artifacts. In case these devices are not available, it is determined that the process is not viable and it ends. 4) From the targeted views, identify the possible techniques and tools needed for the architecture recovery. 5) To check the availability of the selected tools. In case the tools are not available, it is determined that the process is not viable and ends. 6) To identify the need to qualify human talent, or to hire specialized staff, defining the related costs. 7) To calculate the costs of the architecture recovery process and determine its viability based on the tradeoff analysis. The output it generates is the decision that determines the viability of the architecture recovery process, achieving the second milestone of the inception phase. Additionally, the objective views, the list of available and required artifacts, the list of techniques and tools, that are going to be used and the cost of the process are established.

3) *Process planning*: Its goal is to define the work plan, which serves as an orientation to fulfill the proposed objective. It assigns the temporal order of the execution of the activities, the responsible individuals and the deliverables for each activity. Time allocation is made from the sequential order established for the process, considering the experience of the stakeholders involved. It generates as output the work plan, determining times, resources and responsible individuals, fulfilling the third milestone established in the inception phase.

Data Extraction Phase: It establishes as a milestone the identification of the structural and behavioral elements that comprise the software product and the relationships between them at a low level. To achieve this, two activities are defined:

1) *Definition of the domain model of the application*: It is achieved by using conceptual modeling techniques, which require the inputs corresponding to the description of the problem, the knowledge of the expert, the users and the technical staff. It is proposed as the first activity of the data extraction phase. It is likely to be performed iteratively in four moments. The first one corresponds to the definition of the viability of the process, when the required objective views and the artifacts to be recovered are identified, because it serves as a guideline to advice the software engineering expert on these tasks. The second moment is when the process plan is established, because it serves as support to make decisions about the tasks to be performed and to allocate the resources. The third moment occurs during the data extraction phase, because it helps to guide the process, and finally, at the knowledge organization phase, because it can be used as a reference for the abstraction of the models.

2) *Artifact decomposition*: To achieve the representation of the concrete architecture, according to the objective views established at the inception phase, it is necessary to execute transformation processes. Such processes start from the artifacts that represent the implementation of the system. The first transformation consists of decomposing the artifacts. Its purpose is to identify the elements that compose the system and their relationships. This is achieved by applying static and dynamic analysis techniques, according to what is established in the work plan defined for the process. The result of this activity is the system model at a low level, which was obtained using of multiple representations, such as FAMIX, KDM, Rigi Standard Format, among others [8].

Knowledge Organization Phase: It establishes as a milestone the conversion of the system model at a low level into a high-level model. The data obtained from the extraction phase is represented and stored, allowing an easy and efficient recovery, an analysis of the elements that compose the system and their relationships. This phase includes two activities:

1) *Models abstraction*: The large amount of information obtained from the phase of data extraction and the complexity of its representation, makes its interpretation difficult. The goal of this activity is to identify elements and relationships at a high level, based on the system at a low level. This is achieved by applying mapping rules and using the knowledge of the expert in the problem and application domain. The knowledge organization techniques used in this activity can be classified into: manual, automatic and semiautomatic [44].

2) *Models representation*: It aims to represent the elements and relationships identified during the model abstraction, in high-level system models represented in the XML language, under the XMI specification, achieving the milestone established for this phase. This is achieved by applying mapping rules based on the semantic equivalences between the elements and the relationships, which were obtained during the abstraction process, with their equivalents in the unified modeling language.

Information Exploration Phase: Its purpose is to provide the mechanisms to visualize, analyze, navigate and present the results of the architecture recovery process. Another milestone is the construction of the report, which contains the analysis of the recovered architectural views. Therefore, the objectives established for the process at the inception phase are fulfilled. It includes the following activities, that can be executed sequentially or in parallel, depending on the how the group works.

1) *Visualization*: It consists of presenting the results obtained during the knowledge organization phase. Graphics are used for a better understanding and interpretation by the stakeholders. Visualization can be done at the code level, at the class level and at the architecture level, using source code editors and CASE modeling tools. For visualization, some mapping techniques and metaphors must be used to present target views represented in UML, based on the models obtained from the abstraction phase of knowledge represented in XMI. Visualization is complemented with browsing strategies, to make possible the exploration of the information.

2) *Analysis of results*: The interpretation of the obtained models is established as a goal. It applies inference techniques and uses the domain expert's knowledge. It is recommended to use the proposed query mechanism.

3) *The formulation of results*: The organization and recording of results under a readable structure, enables the communication between the stakeholders. The latter serves as support for decision making within the context where the problem arises. During activity the results are organized on a final report, to facilitate their interpretation by each stakeholder. The document is based on the analysis of the results made by the experts in the problem domain and the application.

5.3 Query mechanism

The designed mechanism is a conceptual proposal. It serves as an instrument to support the analysis made by the stakeholders. It offers the possibility of asking questions in natural language to the recovered models. It is limited to UML models and KDM representations in XMI. Therefore, its capability is determined by the query possibilities provided by the XQuery language. The mechanism by itself does not perform any type of analysis. The query mechanism consists of two types of elements (See Figure 5). The first group corresponds to the artifacts that the mechanism receives or generates. The following artifacts are in this group: 1) The original query: it is a question formulated in natural language; 2) The processed query: it is a sentence written in XQuery; 3) The result: it is a text string in natural language, which is the response to the initially formulated question; 4) The UML model: it is an XMI file which contains UML diagrams; and 5) The KDM representation: it is also an XMI file that represents the software product, its elements, associations and execution environments. The last two artifacts are stored in a repository.

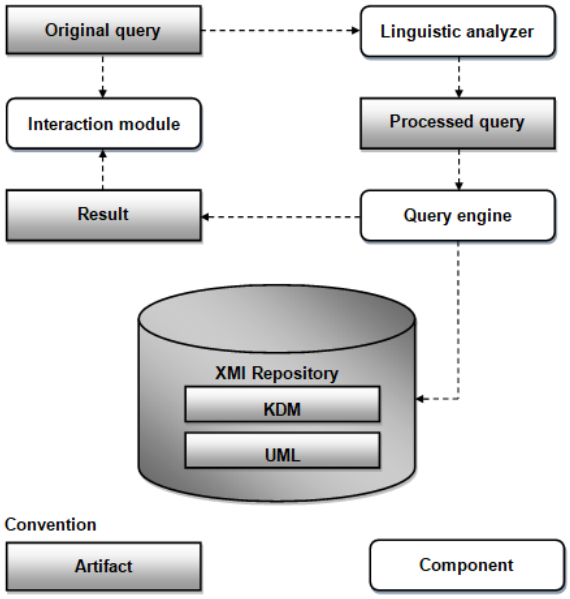

Figure 5. Query mechanism

The second group corresponds to the components that perform a specific task of the mechanism. This group is made up of: 1) the interaction module: it is responsible for capturing the original query and presenting the results; 2) the linguistic analyzer: it verifies the syntax and grammar of the original query to determine if it is accepted by the mechanism. If so, it transforms the original query into processed query. 3) The query engine: it executes the processed query on the repository that contains the models and releases a result.

The expected pattern of the query mechanisms behavior obeys the following algorithm (See Figure 6). The stakeholder writes the query in natural language, which is read by the interaction module. Then the linguistic

analyzer performs the verification of syntax and grammar. If the query is valid and can be processed by the mechanism, the type of query will be identified according to the following classification: 1) metrics calculation, 2) simple query and 3) compound query. Subsequently, the query engine runs it according to the identified type. Finally, the interaction module shows the outcomes of the process. The QModel-XMI tool was implemented as a prototype of the proposed query mechanism. Its main benefit over other similar tools is that QModel-XMI allows the formulation of queries in natural language, namely Spanish.

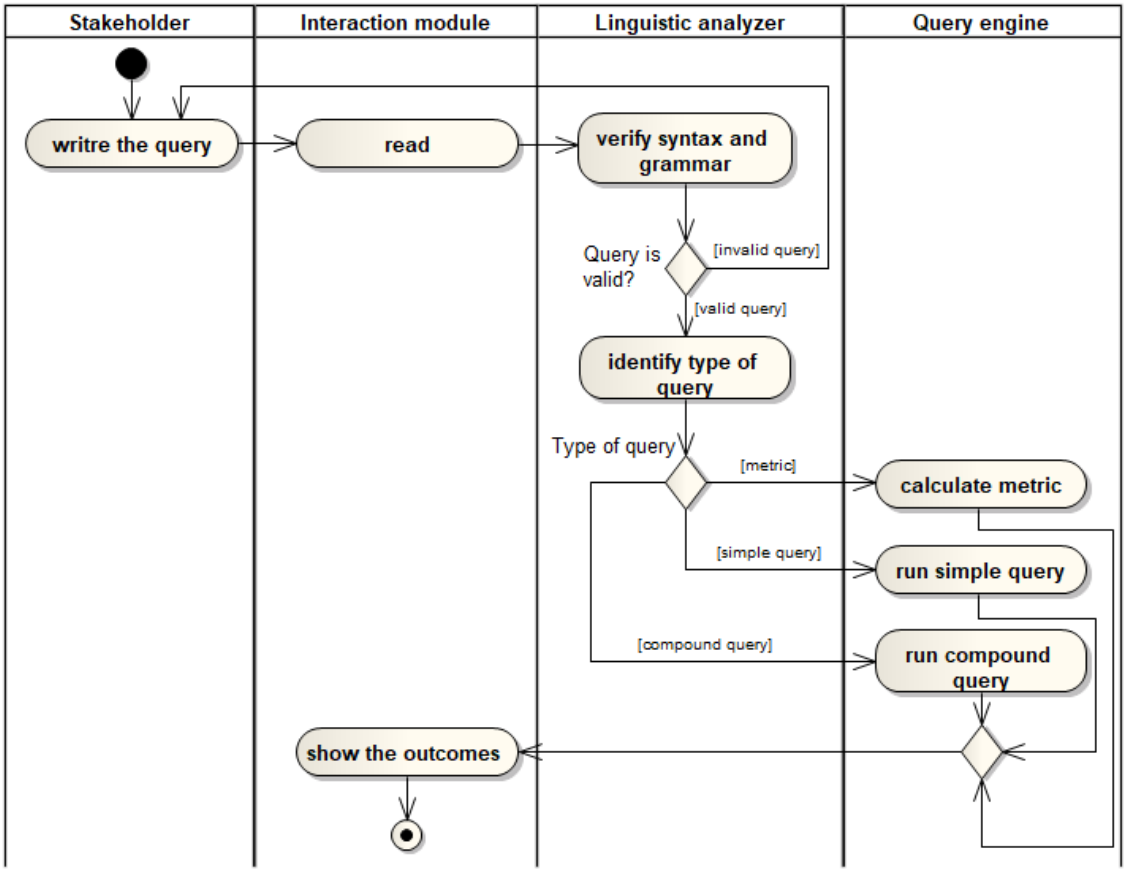

Figure 6. Query mechanism process

5.4 Contexts of use

The following contexts of use of reverse engineering were determined [13]:

1) Software production: It includes all the processes that define the software life cycle, established in the ISO/ IEC 12207: 2008 standard. Reverse engineering is used specifically in the following processes: software maintenance, documentation, software verification and assets reuse.

2) Computer Security: In this field, reverse engineering is used to analyze and understand the malicious code, to define solution strategies and mitigate risks [19][20][21]. It can also be used to identify possible security flaws in software products, as demonstrated in [22].

3) Forensic Computing: In this context, reverse engineering is used to retrieve and present data that has been processed electronically and stored in a computational medium. Therefore, it is possible to interpret them and build evidence to establish the facts, and formulate hypotheses related to the problem under investigation [23].

4) Education: Reverse engineering has been used as a teaching tool that enables [15][16][17][18]: Learning from real successes and failures, the development of curiosity, the development of modeling skills, to program and maintain software products from real implementations. Besides, it facilitates the understanding of concepts in the software engineering field, like it is the case of identifying patterns by applying reverse engineering techniques [4].

VI. EVALUATION

A pattern matching [30] approach is applied to characterize the approaches identified in the literature review. ARCo's structure is defined by applying the logical model approach. The results were evaluated through a single case study in two different contexts. The design of the case study was based on its five components: the research question, the propositions, the units of analysis, the logic to link the data with the propositions and the criteria for interpreting the discoveries [30]. The research question formulated was: How does ARCo contribute to the process of recovery and analysis of architectural views?

The main approach states that ARCo guides the process of recovery and analysis of architectural views, taking into account the particular characteristics of the situation that surrounds the context in which such process is carried out. The first unit of analysis was carried out in the context of education. The purpose was to support the teaching-learning process, in a course with 16 students of Object-Oriented Programming, offered by the Systems Engineering program of the Universidad de Cartagena. The software product under study was JHotDraw version 7.0.6 [52]. The second unit of analysis was performed in the context of software development, at an industrial company in the city of Cartagena (Colombia). The purpose was to recover the documentation of the ICR system used for the control of production and logistics, in order to expand its functionality. We used the analysis based on theoretical propositions as the main strategy to argue the interpretation of the results, and to make logical inferences from the conceptual approaches identified in the literature review. The results of the case study are presented below for each unit of analysis, based on the application of ARCo, highlighting the compliance with the milestones established for each phase. In both contexts we used the following two evaluation criteria: 1) Accuracy and 2) the need of having a thorough knowledge about reverse engineering. Accuracy is measured based on the conformance between expected results and obtained results. The second criterion is valued for each of the stakeholders and measured with the number of stakeholders who are experts in reverse engineering.

6.1 Contexts of education

The learning of object-oriented programming was established as the context during the inception phase. The situation is the teaching-learning process at the Computer Systems Engineering program at Universidad de Cartagena. The purpose of the context is to understand the object-oriented programming paradigm and to develop skills to use it. The objective of the process is to understand the concept of polymorphism and to develop skills to use it properly. The professor who directs the course and the students are the stakeholders. None of them have a thorough knowledge about reverse engineering. The process was determined to be viable because the resources were available: source code of JHotDraw, Enterprise Architect and QModel-XMI. It was established that the views to be recovered are: decomposition view and components and connectors view. In addition, the corresponding work plan was defined.

In the data extraction phase, the domain of the application was defined, establishing that JHotDraw is a Java framework for the creation of structured technical graphics. It provides support for a wide range of programs, from simple drawing style editors, to more complex programs, with rules about how their elements can be used and modified (for example, a tool for creating UML diagrams). It provides support for the creation and edition of geometric shapes defined by the user, establishing behavioral limitations in the editor and the animations. Moreover, the system turned out to be composed of 1,217 classes and 37 interfaces distributed in 122 packages. In total it has 32,054 lines of code and 18,931 lines of comments. As a result, the low-level system model represented in XMI was obtained by using the Enterprise Architect tool.

In the knowledge organization phase, the decomposition view (Figure 7) and the components and connectors view were reconstructed, by using mapping rules, manual and semi-automatic techniques. The elements of the system and their relationships were identified at a high level. Finally, during the exploration phase of information, the students used Enterprise Architect to visualize the results obtained from the process. However, as seen in Figure 7, it is very difficult to visually detail the classes and their relationships. Therefore, the students used QModel-XMI query mechanism (Figure 8) to analyze and interpret the results identifying polymorphic behaviors.

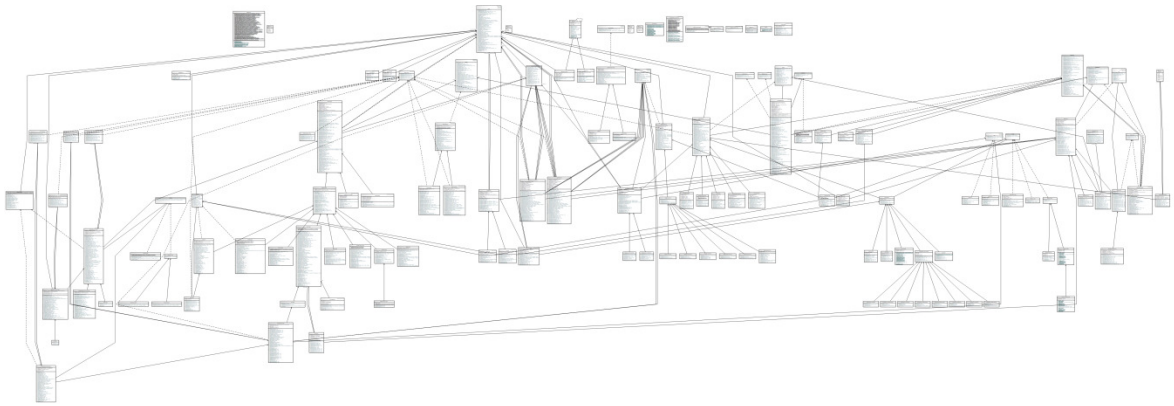

Figure 7. JHotDraw decomposition view

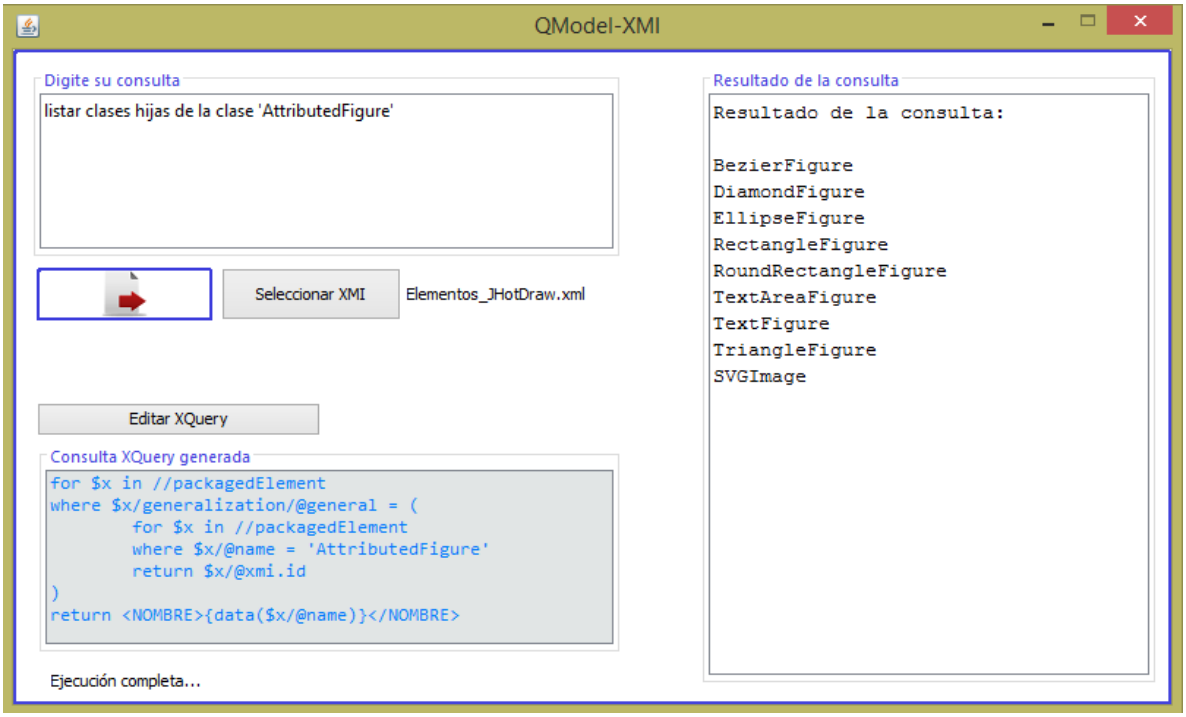

Figure 8. QModel-XMI

Unlike the context of software development process, for this analysis unit, the essence of the results does not lie on the recovered artifacts, but on the achievement of learning objectives. That was possible because the professor used ARCo to design the pedagogical strategy, and so did the students in the development of the class. This was demonstrated in the results of the evaluation made by the professor to the students and whose results were the following: 94% of the students demonstrated that they understood the concept of polymorphism. 84% were able to identify where it is used in the JHotDraw source code using QModel-XMI. 75% developed skills to apply this concept by expanding the functionality of JHotDraw. Besides the academic activity, all of the students who were interviewed, affirmed that the use of QModel-XMI and ARCo stimulated curiosity and motivated them to learn.

6.2 Software development contexts

The construction of a customized software, to support the organizational processes of a company in the industrial sector of Cartagena, was established as the context during the inception phase. The purpose of the context is to improve a software product that supports the tasks associated with production control. The objective of the process is to recover the documentation of the software. Therefore, its functionality can be expanded to new requirements requested by senior management. The stakeholders are: Product architect, system analyst, head of the systems department, expert in the domain of the problem and senior management. None of them have a thorough knowledge

about reverse engineering. The process was determined to be viable because the resources were available: source code of the software product, database dictionary and models of the business processes. The views to be recovered are: decomposition view, layers, classes, components and connectors and the deployment view. The corresponding work plan was also defined.

In the data extraction phase it was defined that the domain of the application named ICR, corresponds to the control of the production of plastic containers. This software allows a control of the production inventory and the raw material used. It also generates key performance indicators which support decisions of the top management. Besides, it was identified that the system is a Java application, consisting of 1,302 classes and 3 interfaces distributed in 173 packages. In total it has 66,095 lines of code and 9,872 lines of comments. For persistence, it uses the database engine: MARIADB, and has interfaces with the SQL Server database server. The Enterprise Architect tool was used to recover the system model at a low level in XMI format.

Mapping rules and manual and semi-automatic techniques were applied to the low-level system model during the knowledge organization phase. The system elements and their relationships were identified with the QModel-XMI tool, obtaining the decomposition views, layers, classes and components and connectors (see Figure 9). The deployment view was obtained manually based on the infrastructure used by the application at the moment the case study was conducted. Finally, the graphical representation of the results, obtained from the process, was achieved, by using models represented in UML with the support of the Enterprise Architect modeling tool during the information exploration phase. The analysis of the results was registered in the report presented to the company. The following aspects are highlighted, without affecting the signed confidentiality agreement: 1) the software product counts on a well-organized solid architectural structure. 2) The use of software development best practices was evidenced. 3) It was recommended to perform a code cloning analysis to debug the system and identify possible reusable assets.

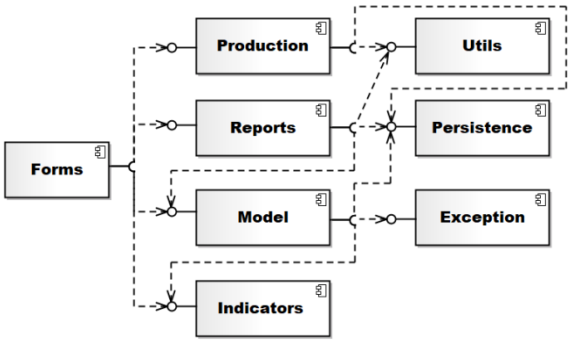

Figure 9. ICR Component and connectors view

VII. ANALYSIS AND DISCUSSION

The analysis of the results is done from two perspectives. Initially it is carried out directly from the results of the case study, and subsequently it is made from the comparison of ARCo with similar approaches.

7.1 Results of the case study

The use of ARCo allowed the achievement of the objectives proposed for each of the architecture recovery processes, for each context of use at the case study taken into account. In the context of education, it supported the learning process in an object-oriented programming course lecture; while in the context of software production, it served to retrieve the documentation of an application used to control the production of plastic containers. The accuracy criterion was achieved in both contexts. In the case study of the context of education, the results obtained were satisfactory because the students understood the concept of polymorphism and developed skills to use it properly. In the case study of software production context, the expected results were also achieved because the decomposition views, layers and components and connectors were recovered. In the same way, the second evaluation criterion was fulfilled in the two case studies, because in both, the stakeholders that participated did not have thorough knowledge about reverse engineering.

In the case of the context of software production, the same results can be obtained by applying the approaches defined in [8][9][10][11][12]. Nevertheless, it is not the same case in the context of education, because these approaches do not take into account the aspects related to the ambit of the context; neither the situations that arise, the available resources, nor the stakeholders, nor their purposes. In the latter case, the main purpose is not the recovery of the architecture, but the learning of the students.

7.2 Comparison of arco with similar approaches

ARCo is obtained as a result of the integration of the existing approaches [2][8][9][10][12][44][46] and it complements them, by including the decisive aspects of the context. This allows the recovery and analysis of software product architectures, to be made according to the stakeholders and their purposes in each context. ARCo also incorporates a conceptual model that facilitates the understanding of reverse engineering. This is useful in the case for those stakeholders who are not experts in that issue. This was evident in both case studies, in which people who do not have thorough knowledge in reverse engineering participated. Likewise, ARCo establishes guidelines and defines a methodology that obeys the generic process [2], where it adds an inception phase to specify the context of the problem, to define the feasibility of performing the process, and to establish the work plan. Additionally, ARCo defines a query mechanism which is implemented in the QModel-XMI prototype, which serves to support the analysis of architectural views represented in UML models. ARCo's main advantage is that it allows to consult in natural language (Spanish). This makes it possible for stakeholders with little experience to do this type of analysis.

A benchmarking of the existing approaches was made. Accuracy and the need of having a thorough knowledge about reverse engineering were used as a criterion, at the moment of comparing ARCo with the approaches identified during the literature review. Accuracy is the ability to achieve the desired objective by the stakeholders involved in the process considering the context. In the specific case of CaCOphoNy [32] it was made for the context of Dassault Systèmes as a software company, and it can be applied in other companies of this type. The approach of Pinzger et al. [45] also applies to the context of software production, specifically for the product families' development. The same happens with REM [7], QADSAR [3] and the approach of Boussaidi et al. [8] oriented to the modernization of a large legacy system.

Within the same context, are the approaches of García et al [12], who propose a framework for Obtaining the Ground-Truth in Architecture Recovery; and the approach of Sartipi et al. [9], who propose an orchestrated set of techniques and a multi-view toolkit to reconstruct three views of a software system, such as design, behavior, and structure. These last two approaches are focused on the software product maintenance. On the other hand, Symphony [44] poses a generic process based on views to reconstruct software architecture. All of these approaches assume that the architecture recovery process is performed by experts in their respective domain. Therefore, they allow to achieve the desired objective of the stakeholders that participate in the process, within the context of software production. However, this does not happen when one of these approaches is applied to a context such as education. In that case the stakeholders were students, who are learning software engineering; and professors, who are experts in software engineering, but do not have expertise in reverse engineering nor in architecture recovery. That is the benefit of ARCo compared to existing approaches. As shown in the results of both case studies, ARCo allows engineers / students to reconstruct / understand an architecture in a more accurate way. The students, with minimal knowledge about reverse engineering, performed an architecture recovery of JHotDraw and achieved meaningful results.

7.2 Lessons learned

The first lesson learned confirms the following two statements: 1) We confirm the thesis of Favre [32] that: "software architecture is in practice much more complex than the academic world tends to explain". In addition, we found out that the process of recovery and analysis of architectural views should be light and flexible, so it can be understood and applied by those who are interested in the different contexts of use. 2) We also confirm the statement of Jansen et al. [53] that: "Architectural views are subjective views", because in each context where the recovery process was applied, the details of the architecture recovered views were defined by the stakeholders.

From the case studies, we learned: 1) In order to lead an architecture recovery process, it is not enough to have technical skills, because the subjectivity of the stakeholders involved in the process, demand special skills to balance interests and to guide them towards meeting the objective. 2) The achievement of the objectives of the process greatly depends on the ability to determine the scope and resources of each situation when recovering and analyzing the architectural views.

VIII.CONCLUSION

In this work, we presented ARCo, a generic framework for architecture recover that considers the particular characteristics of the situation that surrounds the context where such process takes place. We showcased ARCo with two case studies, each from a different context. Our preliminary results show that 1) The complexity of the architectural views recovery process must be hidden, so that it can be applied in different contexts with the participation of stakeholders who are not experts in reverse engineering; 2) Unlike the existing approaches, ARCo makes possible the recovery and analysis of architectural views, in a suitable way with to context where the process is

performed; 3) ARCo integrates existing approaches and complements them by including the context, a conceptual model, a set of guidelines, a methodology and a query mechanism; and 4) ARCo does not define new techniques, nor does it modify the canonical process for the recovery and analysis of architectural views. However, it does offer a new approach for software architecture recovery, since it involves the context where the process is carried on, and hide its complexity from stakeholders.

For future work, we plan to evaluate ARCo in other contexts, such as computer security and forensic computing. Furthermore, we plan to improve the QModel-XMI tool, for instance, by incorporating artificial intelligence techniques to accept more complex questions formulated in different natural languages.

ACKNOWLEDGMENTS

This work was supported by the Universidad de Cartagena and Universidad del Cauca. We thank to Ismael Sayas Arrieta, Alexander Silvera, Jesus David Prasca and Esteban Triviño for develop source code of QModel-XMI.

REFERENCES

[1] Monroy, Martín E., José L. Arciniegas, and Julio C. Rodríguez. "Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura." Información Tecnológica 27.5 (2016): 201-220.

[2] Tilley, Scott R., Santanu Paul, and Dennis B. Smith. "Towards a framework for program understanding." Program Comprehension, 1996, Proceedings., Fourth Workshop on. IEEE, 1996.

[3] Stoermer, Christoph, O. Liam, and Chris Verhoef. "Moving towards quality attribute driven software architecture reconstruction." null. IEEE, 2003. p 46.

[4] Stormer, C. Software quality attribute analysis by architecture reconstruction (squa3re). In Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on (pp. 361-364). IEEE.

[5] Huang, Gang, Hong Mei, and Fu-Qing Yang. "Runtime recovery and manipulation of software architecture of component-based systems." Automated Software Engineering 13.2 (2006): 257-281.

[6] Schmidt, Frederik, Stephen G. MacDonell, and Andrew M. Connor. "An automatic architecture reconstruction and refactoring framework." Software Engineering Research, Management and Applications 2011. Springer, Berlin, Heidelberg, 2012. 95-111.

[7] Tamburri, D. A., & Kazman, R. (2018). General methods for software architecture recovery: a potential approach and its evaluation. Empirical Software Engineering, 23(3), 1457-1489.

[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.

[9] Sartipi, Kamran, Nima Dezhkam, and Hossein Safyallah. "An Orchestrated Multi-view Software Architecture Reconstruction Environment." WCRE. 2006.

[10] Yan, Hong, et al. "Discotect: A system for discovering architectures from running systems." Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on. IEEE, 2004.

[11] Callo Arias, Trosky B., Pierre America, and Paris Avgeriou. "A top‐down approach to construct execution views of a large software‐intensive system." Journal of Software: Evolution and Process 25.3 (2013): 233-260.

[12] Garcia, Joshua, et al. "A Framework for Obtaining the Ground-Truth in Architectural Recovery." WICSA/ECSA. 2012.

[13] Monroy, Martín E., José L. Arciniegas, and Julio C. Rodríguez. "Caracterización de los Contextos de Uso de la Ingeniería Inversa." Información tecnológica 28.4 (2017): 75-84.

[14] Tonella, Paolo, et al. "Empirical studies in reverse engineering: state of the art and future trends." Empirical Software Engineering 12.5 (2007): 551-571.

[15] Klimek, I., Keltika, M., & Jakab, F. (2011, October). Reverse engineering as an education tool in computer science. In Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on (pp. 123-126). IEEE.

[16] Cipresso T., Software reverse engineering education. 2009.

[17] Ali, M. R. (2005). Why teach reverse engineering? ACM SIGSOFT Software Engineering Notes, 30(4), 1-4.

[18] Ali, M. R. (2006). Imparting effective software engineering education. ACM SIGSOFT Software Engineering Notes, 31(4), 1-3

[19] Treude C., Figueira F. and M. Storey. An Exploratory Study of Software Reverse Engineering in a Security Context. 18th Working Conference on Reverse Engineering, Limerick, Ireland, pp. 184 – 188. 2011.

[20] Ligh, M., Adair, S., Hartstein, B., & Richard, M. (2010). Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code. Wiley Publishing.

[21] Sharif, M., Lanzi, A., Giffin, J., & Lee, W. (2009, May). Automatic reverse engineering of malware emulators. In Security and Privacy, 2009 30th IEEE Symposium on (pp. 94-109). IEEE.

[22] Abi-Antoun, M., & Barnes, J. M. (2010). Analyzing security architectures. Paper presented at the ASE'10 - Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, 3-12.

[23] Nelson, B., Phillips, A., & Steuart, C. (2015). Guide to computer forensics and investigations. Cengage Learning.

[24] Zhong, L., Ye, H., & Xia, J. (2018, November). Research on Software Evolution Reconstruction Based on Architecture Recovery. In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) (pp. 68-71). IEEE.

[25] Ducasse, Stephane, and Damien Pollet. "Software architecture reconstruction: A process-oriented taxonomy." IEEE Transactions on Software Engineering 35.4 (2009): 573-591.

[26] ISO/IEC/IEEE 42010. Systems and software engineering — Architecture description, International Organization for Standardization, Ginebra, Suiza (2011).

[27] Hofmeister, C.; Nord, R, y D. Soni, Applied Software Architecture, Reading, MA: Addison-Wesley (1999).

[28] 61. Kruchten, P. B. (1995). The 4+ 1 view model of architecture. IEEE software, 12(6), 42-50.

[29] Smolander, K. (2002). Four metaphors of architecture in software organizations: finding out the meaning of architecture in practice. In Proceedings International Symposium on Empirical Software Engineering (pp. 211-221). IEEE.

[30] Yin RK. Case study research: Design and methods. Sage publications; 2013.

[31] Risi, Michele, Giuseppe Scanniello, and Genoveffa Tortora. "Using fold-in and fold-out in the architecture recovery of software systems." Formal Aspects of Computing 24.3 (2012): 307-330.

[32] Favre, J-M. "Cacophony: Metamodel-driven software architecture reconstruction." Reverse Engineering, 2004. Proceedings. 11th Working Conference on. IEEE, 2004.

[33] Garcia, Joshua, et al. "Enhancing architectural recovery using concerns." Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on. IEEE, 2011.

[34] Cai, Yuanfang, et al. "Leveraging design rules to improve software architecture recovery." Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures. ACM, 2013.

[35] Kobayashi, Kenichi, et al. "SArF map: Visualizing software architecture from feature and layer viewpoints." Program Comprehension (ICPC), 2013 IEEE 21st International Conference on. IEEE, 2013.

[36] Huang, Gang, Hong Mei, and Fu-Qing Yang. "Runtime recovery and manipulation of software architecture of component-based systems." Automated Software Engineering 13.2 (2006): 257-281.

[37] Kang, Sungwon, Seonah Lee, and Danhyung Lee. "A framework for tool-based software architecture reconstruction." International Journal of Software Engineering and Knowledge Engineering19.02 (2009): 283-305.

[38] Pollet, Damien, et al. "Towards a process-oriented software architecture reconstruction taxonomy." Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on. IEEE, 2007.

[39] Vasconcelos A, Werner C. Evaluating reuse and program understanding in ArchMine architecture recovery approach. Information Sciences. 2011; 181(13):2761–86.

[40] Kazman R, O'Brien L, Verhoef C. Architecture reconstruction guidelines. Third Edition. Carnegie Mellon University. Software Engineering Institute, Pittsburgh. 2003; p. 1–43.

[41] Monroy, Martín E., José L. Arciniegas, and Julio C. Rodríguez. "Caracterización de herramientas de ingeniería inversa." Información tecnológica 23.6 (2012): 31-42.

[42] Bruneliere, H., Cabot, J., Dupé, G., & Madiot, F. (2014). Modisco: A model driven reverse engineering framework. Information and Software Technology, 56(8), 1012-1032.

[43] Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., & Di Salle, A. (2017, April). Towards recovering the software architecture of microservice-based systems. In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW) (pp. 46-53). IEEE.

[44] Van Deursen, Arie, et al. "Symphony: View-driven software architecture reconstruction." Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on. IEEE, 2004

[45] Pinzger, Martin, et al. "Architecture recovery for product families." International Workshop on Software Product-Family Engineering. Springer, Berlin, Heidelberg, 2003.

[46] Riva, Claudio. "Architecture reconstruction in practice." Software Architecture. Springer, Boston, MA, 2002. 159-173.

[47] Moreira MA, Greca I, and M. L. R. Palmero. "Mental models and conceptual models in the teaching and learning of science." Revista Brasileira de Investigação em Educação em Ciências 2.3 (2002): 84-96.

[48] ISO/IEC 19506:2012. Information technology -- Object Management Group Architecture-Driven Modernization (ADM) -- Knowledge Discovery Meta-Model (KDM).

[49] Gall H et al. "Architecture recovery in ARES." Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints' 96) on SIGSOFT'96 workshops. ACM, 1996.

[50] McGregor, Sue LT, and Jennifer A. Murnane. "Paradigm, methodology and method: Intellectual integrity in consumer scholarship." International journal of consumer studies 34.4 (2010): 419-427.

[51] Monroy M, Rodríguez J, Puello P. Characterization Model of Software Architectures Recovery Process. Indian Journal of Science and Technology. 2018; 11(1):1–10.

[52] Gamma E and T. Eggenschwiler. JHotDraw as Open-Source Project. https://www.jhotdraw.org/

[53] Jansen, A., Bosch, J., & Avgeriou, P. (2008). Documenting after the fact: Recovering architectural design decisions. Journal of Systems and Software, 81(4), 536-557.