Vol 26 (1)

Ingeniería y
Competitividad
Vol 26 (1) 2024; doi: 10.25100/iyc.v26i1.12840

ISSN
0123-3033
e- 2027-8284

Article of investigation

Applied reverse engineering in context

Ingeniería inversa aplicada en función del contexto

Martín E. Monroy[1] Martin Pinzger[2] José L. Arciniegas[3]

[1]Universidad de Cartagena, Programa de Ingeniería de Sistemas, Cartagena, Colombia
[2]Alpen-Adria-Universität, Software Engineering Research Group, Klagenfurt, Austria,
[3]Universidad del Cauca, Departamento de Telemática, Popayán, Colombia.

Abstract

Reverse engineering is applied in multiple contexts. Each context is defined by a group of stakeholders, a set of resources and situations within a specific scope. There are diverse approaches for reverse engineering, however, all they assume that it is done in the context of software production. The aim of this work is to define an approach to recover the design of software products in different contexts. A comparative analysis of reverse engineering approaches was made using the pattern matching technique. To validate obtained results, a case study was carried out in two distinct contexts, the first in an education context to support a teaching-learning process and the second in a software production context to retrieve a software product design. A framework was defined, which includes a descriptive conceptual system and a set of instrumental elements of operational type, which serves to guide the software product design recovery process, based on the context in which this activity is carried out. It is concluded that the defined framework offers a new approach to software design recovery, because it involves the context where the process takes place and hides its complexity from non-expert stakeholders in reverse engineering.

Resumen

La ingeniería inversa se realiza en múltiples contextos. Cada contexto está definido por un grupo de participantes, un conjunto de recursos y situaciones que se encuentran dentro de un ámbito específico. Existen múltiples propuestas para realizar ingeniería inversa, sin embargo, todas asumen que se hace en el contexto de la producción de software. El objetivo de este trabajo es proponer un referente para recuperar el diseño de productos software, que pueda ser utilizado en diferentes contextos. Se hizo un análisis comparativo de los enfoques de ingeniería inversa utilizando la técnica de coincidencia de patrones. Para validar los resultados obtenidos se realizó un estudio de caso en dos contextos diferentes, el primero en un contexto de educación para apoyar un proceso de enseñanza aprendizaje y el segundo en un contexto de producción para recuperar el diseño de un producto software. Se definió un marco de referencia conformado por un sistema conceptual descriptivo y un conjunto de elementos instrumentales de tipo operativo, que guía el proceso de recuperación del diseño de productos software, ajustándose a las características del contexto en el que se realiza esta actividad. Se concluye que el marco de referencia definido, ofrece un nuevo enfoque para la recuperación del diseño de productos software, porque involucra el contexto en el que se realiza el proceso y oculta su complejidad a los participantes que no son expertos en ingeniería inversa.

Keywords: Design recovery, Education, Framework, Reverse engineering, Software production.

Palabras clave: Educación, Ingeniería inversa, Marco de referencia, Producción de software, Recuperación del diseño.

How to cite?

Monroy, M.E., Pinzger, M., Arciniegas, J.L. Applied Reverse Engineering in Context. Ingeniería y Competitividad, 2024, 26(1); e-22112840.

https://doi.org/10.25100/iyc.v26i1.12840

Recibido: 2-03-23
Aceptado: 12-02-23

Correspondencia:
mmonroyr@unicartagena.edu.co

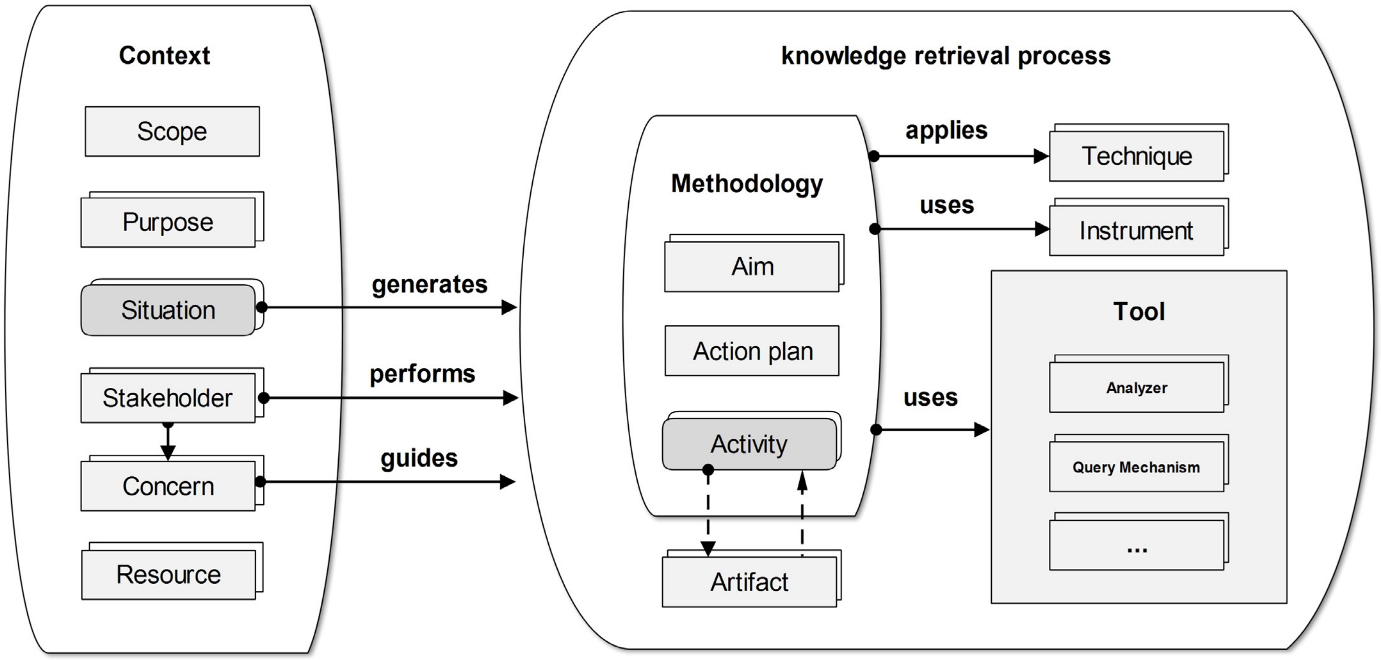
Conflict of interest: none declared

Why was it carried out?
Reverse engineering is applied in multiple contexts. Each context is defined by a group of stakeholders, a set of resources and situations within a specific scope. There are diverse approaches for reverse engineering, however, all they assume that it is done in the context of software production. The aim of this work is to define an approach to recover the software design according to the context situations and stakeholders' concerns.

What were the most relevant results?
A framework was defined, which includes a descriptive conceptual system and a set of operational type instrumental elements, which serves to guide the software product design recovery process, based on the context in which this activity is carried out.

What do these results provide?
A conceptual model to recover the design and to analyze the recovered documentation, based on the ISO/IEC/IEEE 42010, ISO/IEC 19506 and UML standards.
A methodology that guides the reverse engineering process, according to the context where the need arises, and that allows stakeholders to obtain relevant results to them.
The characterization of the reverse engineering contexts of use.
The inclusion of context analysis in the reverse engineering process, represented by the scope, situations, resources, stakeholders and their concerns.
The design and construction of a query mechanism prototype to support the recovered documentation analysis.

Graphical Abstract



doi: 10.25100/iyc.v26i1.12840

Introduction

There are multiple approaches to recover the design of software products, classified into techniques (1-4), tools (6-10), methods (11-13), and frameworks (5, 14-18). Techniques detail how a specific activity of the reverse engineering process should be carried out. Methods focus on the process and define the logical order of required activities to recover the design. Tools implement techniques to automate the process, and frameworks encompass methods, techniques, and tools. The identified approaches apply activities defined by Tilley et al. (19): data extraction, knowledge organization, and information exploration. Some focus on software product analysis (11), others on supporting evolution (12,20), on design reconstructing (12-16), or on product redesign (17). All are oriented towards the software engineering process, for maintenance, quality control, redesign, and asset reuse (21).

Reverse engineering is a research field that has attracted the attention of researchers in recent years from multiple perspectives (12,22,23). Its main goal is to reconstruct the implicit knowledge in a system, represented in its architectural views (5). Software architecture is the fundamental organization of a system embodied in its components, the relationships that exist between them and their environment, and the principles that govern its design and evolution (24). An architectural view expresses the system's architecture through models, built from the perspective of constraints defined by stakeholders (24). There are various approaches that specify architectural views and software viewpoints. This implies selecting an approach to represent system views that fits the situation and context, which increases the complexity of the recovery process because it requires considering situations, available resources, stakeholders, their limitations, and purposes.

The reverse engineering study has focused on defining techniques specialized in reconstructing system design (15,17), in product review (25), evolution and analysis (11), or in achieving one or more of these purposes simultaneously (12). These approaches focus more on specific activities of reverse engineering than on the process itself. The canonical process defined by Tilley et al. (19) has been refined by several proposals. For example, Symphony (18) focuses on general aspects of architecture recovery and how to select views to reconstruct. Cacophony (17) defines a generic process driven by meta-models. Kerdoudi et al. (13) define a product lines design recovery process. Tamburri and Kazman (12) propose a generic method for architecture recovery. Stormer (11) focuses on architecture recovery from the perspective of quality attributes. Bruneliere et al. (5) define a framework based on KDM (Knowledge Discovery Metamodel) aimed at product modernization. Tamburri and Kazman (12) suggest a process to extract three system views to facilitate maintenance; and Schmitt et al. (15) propose a reference framework that includes a set of principles and processes for recovering architectures. Each of these approaches defines a set of activities for recovering design and establishes the logical order under which they should be performed.

On the other hand, Stormer (11) identifies the contexts of reverse engineering as: improving the understanding of the legacy software systems architecture, enhancing the architecture itself, evaluating the quality system attribute characteristics, and improving the system design. All of these activities are typical in the software construction process. For Favre (17) and Ibrahim et al. (14), the context refers to the physical environment or situation of the application, while for Tamburri and Kazman

doi: 10.25100/iyc.v26i1.12840

(12), the context corresponds to the development circumstances of the software product. Only Van Deursen et al. (18) consider aspects related to resource availability and the stakeholders concerns when recovering the design of the software product. However, none of the identified proposals in the literature review explicitly take into account the situations, stakeholders characteristics, scope, or purpose of the context in which the reverse engineering process is carried out. This creates a gap that complicates the design recovery in contexts other than software production, which becomes the main contribution of this research work.

Reverse engineering is an activity that is also conducted in other contexts such as education (26-28), computer security (25, 29-31), and computer forensics (32), in addition to software production. Each of these contexts is determined by a scope, situations, resources, purposes, and stakeholders with particular concerns and characteristics (23). In this scenario, the following questions arise when reverse engineering is conducted in contexts other than software production: How to tailor the reverse engineering process to be relevant to the characteristics of the stakeholders and their concerns? How to tailor the reverse engineering process based on the situations present in the context where it is conducted? How to tailor the reverse engineering process based on the scope and purpose of the context where it is conducted? And how to conceal the complexity of the reverse engineering process for those stakeholders who are not experts?

Consequently, we propose a reference framework for design recovery and documentation retrieved analysis, involving the stakeholders, their concerns, circumstances related to resource availability, and the situations encountered when conducting a reverse engineering process. The main contributions of this work are: A conceptual model for design recovery and the documentation retrieved analysis, based on ISO/IEC/IEEE 42010, ISO/IEC 19506, and Unified Modeling Language (UML) standards. A methodology that guides the reverse engineering process according to the context in which the need arises, enabling stakeholders to obtain results tailored to their concerns. The characterization of the contexts in which reverse engineering is applied. The inclusion of the context analysis in reverse engineering process, represented by the scope, situations, resources, stakeholders, and their concerns. The design and construction of a prototype query mechanism to facilitate the analysis of the recovered documentation.

Methodology

Qualitative research consisting of four sequential phases was conducted. The first phase involved a literature review following the methodological proposal by Peters et al. (33) to identify reverse engineering approaches used for software product design recovery. The search was conducted in IEEE, ACM, and Scopus databases using the search strings (design OR architecture) AND (retrieval OR reconstruction OR recovery). Only publications from 2018 onwards from indexed journals and conference proceedings in English were included. Duplicate documents and those that did not refer to software reverse engineering were excluded.

In the second phase, the identified proposals were characterized using the comparative analysis technique. The following characteristics were defined to establish similarities and differences: type of proposal, purpose, stakeholders, and the context for which it was created. In the third phase, the conceptual structure of the reference framework

doi: 10.25100/iyc.v26i1.12840

was defined using the logical modeling technique for knowledge representation. Each of the framework components was specified using the pattern matching technique (34) applied to the proposals found in the literature review. In the fourth phase, the reference framework was evaluated. The effectiveness and relevance of the reference framework were assessed through a case study, while its contribution was determined using a strategy involving the examination of possible rival explanations (34). For this purpose, a comparative analysis was conducted between the defined reference framework and similar proposals identified in the literature review. The comparison variables included the activities performed for design recovery, the underlying conceptual model, and the elements constituting the context (purpose, scope, situations, resources, stakeholders, and their concerns).

The case study was conducted based on its five components (34):

Research question: How does the proposed reference framework contribute to the software design recovery process?
The main proposition states that: the proposed reference framework guides the software design recovery process based on the context in which it is applied.
Two units of analysis. One in the context of education and the other in the software development context.
To make logical inferences, effectiveness is defined as the ability of the reference framework to achieve the proposed objective in each unit of analysis, and relevance as the degree to which the reference framework conceals complexity and takes into account the context: scope, purpose, situations, goals, available resources, and stakeholders' concerns (See Table 1).

Criteria for interpreting the results: The reference framework is considered effective if its assessment is above 75% in each unit of analysis of the case study. Relevance is assessed using the Likert scale, based on stakeholders' perception in the case study.

In Table 1, data collection instruments, used resources, and evaluated variables with their respective formulas and metrics are presented.

doi: 10.25100/iyc.v26i1.12840

Table 1. Methodological resources

Criterion		Context	
		Education	Software engineering
Information collection instruments		Student survey	Stakeholders survey
		Teacher interview	Software Architect interview
		Students laboratory reports	Technical report outlining the findings presented to the company
		Report on the academic activity conducted by the teacher	
		Recovered artifacts	
Resources	The subject of study:	JHotDraw version 7.0.6	CPL System version 1.0
	Enterprise Architect	Tools for recovering and visualizing models	
	QModel-XMI	recovering and visualizing models tools	
	Imagix 4D	Source code analysis tool	
Evaluated variables		Equation (1) Effectiveness = (AR*100)/RP	AR: Achieved Result, ER: Expected result
		AR: Degree of achievement of the activity’s objective by the students.	AR: Degree of compliance with the purpose of the recovered design views.
		ER: All students fully achieve the objectives of the activity	RP: All recovered design views fulfill their purpose
		Relevance: The Likert scale was used to quantify participants’ perception regarding the following aspects. Hide the complexity HC, Takes into account the scope Sc, the purpose P, la situation S, the aims A, the resources R, the stakeholders concerns C.	

Results and discussion

Initially, we introduce the reference framework, consisting of a conceptual system and an instrumental part, which emerge from the theory regarding the nature of the reverse engineering process and how to approach it. Subsequently, we explain the evaluation results of the reference framework, and finally, we address the analysis of the results.

Conceptual system

The conceptual system of the reference framework is composed of a conceptual model, a theoretical base, and a conceptual base (see Figure 1). This last component is understood as a scientifically shared, external, precise, comprehensive, and consistent representation of knowledge that facilitates the teaching and understanding of the object of study. The conceptual base and the theoretical base correspond to the available literature on reverse engineering. The proposed conceptual model provides a comprehensive view of reverse engineering and detailed recovery of design and analysis of the retrieved documentation, which facilitates the understanding and study of this field of knowledge.

doi: 10.25100/iyc.v26i1.12840

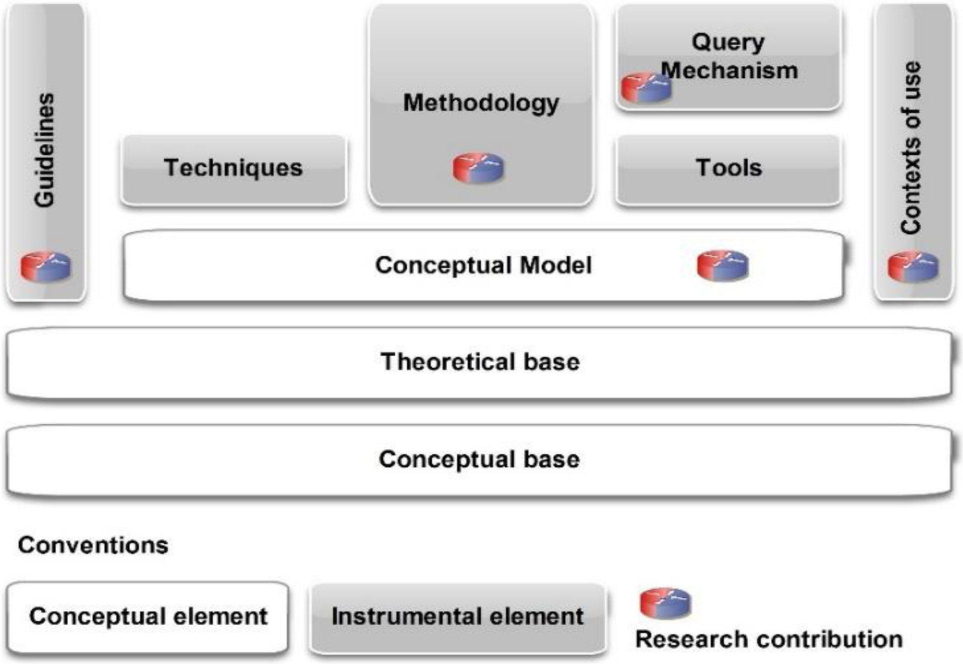


Figure 1. Reference framework

The conceptual model encompasses two aspects: the holistic view of reverse engineering (see Figure 2) and design recovery. Reverse engineering addresses methods aimed at recovering implicit knowledge within software to support the execution of activities requiring an understanding of said system. Therefore, the context in which the need to recover knowledge arises and the process undertaken to achieve knowledge recovery must be considered. Hence, the conceptual model encompasses these two elements, as depicted in Figure 2. The context is determined by the scope, purposes, situations encountered, available resources, and stakeholders with their concerns, as detailed by Monroy et al. (23). Furthermore, the knowledge recovery process is determined by a methodology that utilizes techniques, instruments, and tools to meet established objectives, carrying out activities that generate and require artifacts according to the action plan.

The conceptual model focuses on understanding each activity involved in the canonical process of reverse engineering (19) and the artifacts required or generated in the development of these activities, as observed in Figure 3. The software product and expert knowledge are used as inputs to identify the artifacts constituting the system. These artifacts are decomposed using data extraction techniques. As a result, implicit knowledge within the system is obtained, represented in KDM to facilitate interoperability between tools. The elements explicit in the syntax of the studied product (source packages and KDM code) are obtained by applying mapping rules to the programming language. This is not the case with the abstraction layer and the runtime resource layer because the elements are implicit and correspond to higher levels of abstraction, requiring incremental analysis based on primitive KDM representations.

doi: 10.25100/iyc.v26i1.12840

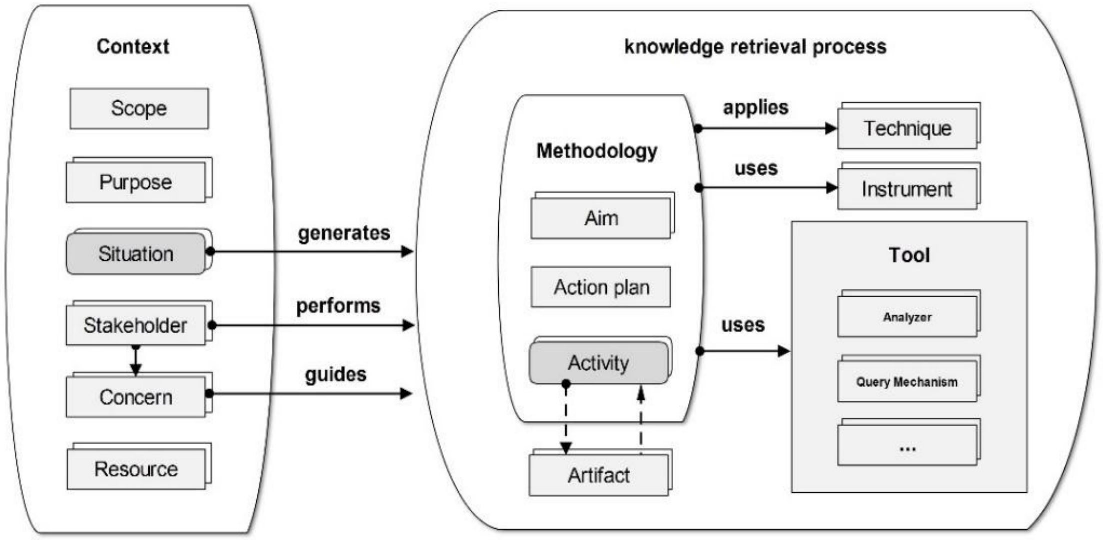


Figure 2. Holistic view of the reference framework.

Knowledge organization techniques are applied to the identified elements to create UML models of the system, represented in XMI to ensure interoperability. Considering the established aims for the reverse engineering process, system views are defined, which may consist of one or several models. Any tool capable of interpreting XMI code is used to visualize the UML models.

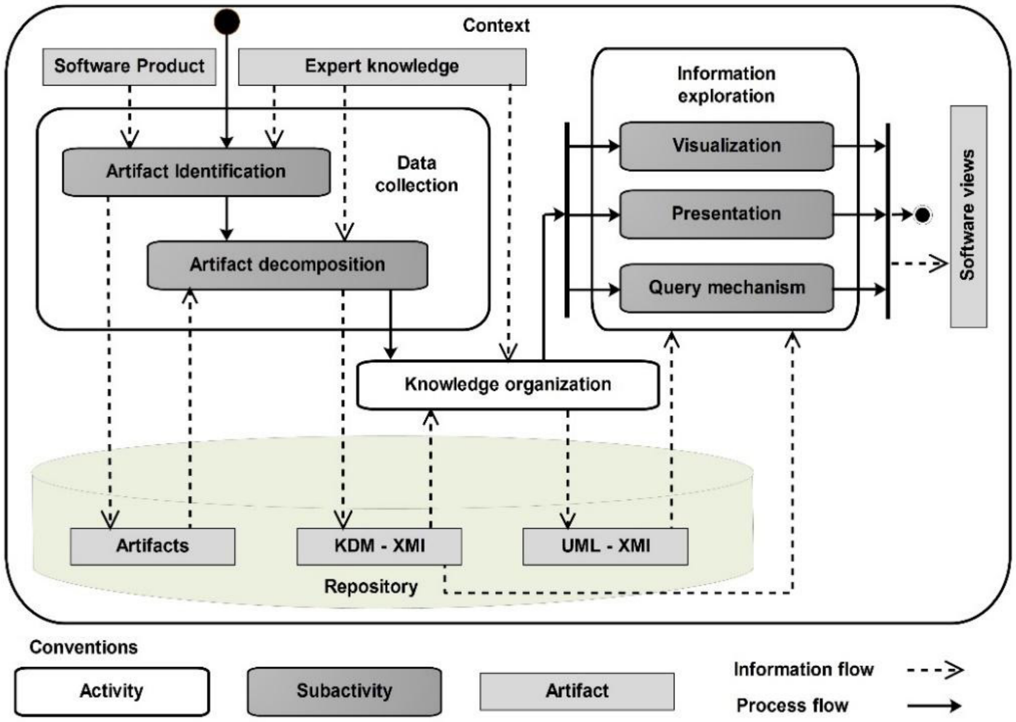


Figure 3. Conceptual model

Each activity has an aim, receives inputs, employs techniques, and produces outputs. The reverse engineering process begins with data extraction (19). Its main purpose is to gather the necessary data to obtain the views to be recovered from the system.

doi: 10.25100/iyc.v26i1.12840

This requires the software product to be analyzed and expert knowledge as inputs. Data extraction includes two tasks: in the first one, artifacts that make up the system are identified, applying techniques such as manual inspection. The result is a set of repositories containing: executables, source code files, binary files, configuration files, resource descriptors, images, documentation, etc. This depends on the availability of such artifacts in the context in which the reverse engineering process is carried out. In the second task, the identified artifacts are decomposed using techniques such as grammatical islands, syntactic analysis, lexical analysis, fuzzy analysis, data flow analysis, syntactic matches, profiling, and code instrumentation. The result is an XMI repository of the KDM representation of the product.

The reverse engineering process continues with the knowledge organization activity, whose purpose is to transform the data derived from extraction to facilitate its storage, retrieval, and analysis. The result is UML models in XMI of the recovered documentation. This is achieved using techniques such as clustering algorithms, SQL queries, Tarski algebra, propositional algebra, relational algebra, reflection model, and ad-hoc query languages. Finally, information exploration is performed. The purpose of this activity is to provide mechanisms to visualize, present the results of the reverse engineering process, and facilitate its analysis. This is achieved with a query mechanism that extracts implicit information in the recovered models, facilitating the understanding of the software product.

Instrumental part

It consists of: the guidelines that establish the directives under which the reference framework was defined and how it should be used; the methodology that guides the reverse engineering process considering the stakeholders, their concerns, and the context in which it takes place; the query mechanism to support the analysis of the recovered design; and the characterization of the contexts in which the reverse engineering process is carried out.

Guidelines

The reference framework was defined based on the guidelines outlined in Table 2. To achieve efficient use of the reference framework, it is suggested to: understand the conceptual elements comprising it, comprehend its conceptual model, utilize the proposed methodology by applying the recommended techniques and tools for each activity, involve an expert in the domain of the application and the problem, and engage a person knowledgeable about reverse engineering instruments, tools, and techniques.

doi: 10.25100/iyc.v26i1.12840

Table 2. Reference framework guidelines

Criterion	Guideline
Aim	To guide the reverse engineering process based on the context in which it occurs, to ensure relevant, accurate, and efficient results, taking into account that some participants lack experience in reverse engineering.
Scope	It encompasses the recovery of the design of object-oriented software systems, irrespective of the technologies used for system development and deployment.
Reference standards	ISO/IEC/IEEE 42010:2022 for architectural description. ISO/IEC 19506:2012 as a meta-model for knowledge discovery. OMG Specification: XMI 2.5.1:2015 y UML 2.5:2015 to facilitate metadata exchange..
References	Technological: The use of programming paradigms, architectural patterns, specific technologies, etc.. Human: It should align with the knowledge, skills, abilities, and interests of the participants in the reverse engineering process. Necessary and available resources for carrying out the reverse engineering process. The context in which the reverse engineering process takes place.

Design recovery Methodology

The proposal is presented as a methodology in coherence with McGregor's four axioms (35). The epistemological axiom is embedded in the conceptual elements of the reference framework. The axiological axiom establishes the following methodological guidelines: 1) The methodology corresponds to the reverse engineering process. 2) Existing proposals are integrated to ensure that the results are appropriate for the context and situation where the reverse engineering process takes place. 3) The reverse engineering process is structured based on the activities defined by Tilley et al. (19). 4) The process defines the proposed goals, activities, and the logical sequence in which they are executed; the input and output resources; as well as the techniques applied and the instruments used to achieve the stated goals. The ontological axiom establishes reverse engineering as the object of study under a specific context (23). Lastly, the logical axiom consists of the sequence of actions that make the reverse engineering process feasible.

In addition to the axioms, the methodology defines a structured process in phases: Inception, data extraction, knowledge organization, and information exploration. Each phase encompasses activities aimed at achieving the goals and milestones set for accomplishing the objectives established for the process, supported by techniques and the use of various tools and instruments (see Figure 4). The sequence of activities is defined in the action plan, which, like the process objectives, is established based on the context and situation in which the reverse engineering process takes place. For each activity, the motivation, stakeholders, and artifacts (inputs and outputs) are defined. If the activity is structured into tasks, these aspects are also defined for each task.

doi: 10.25100/iyc.v26i1.12840

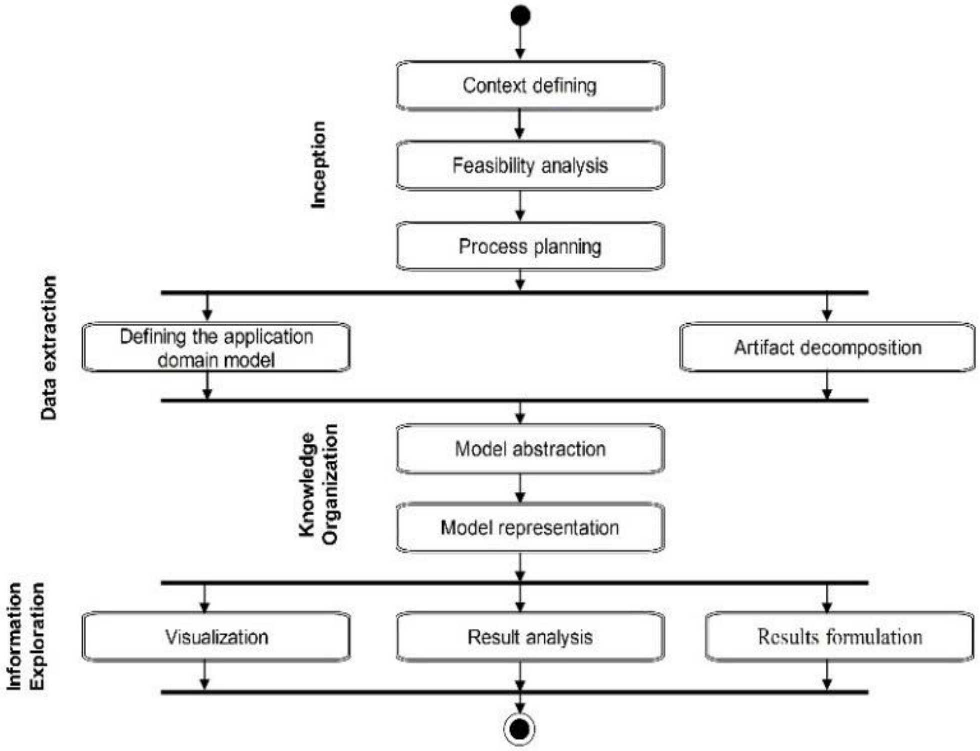


Figure 4. Design recovery methodology.

The inception phase purpose is to understand the problem context, for which the following are identified: the scope, the stakeholders and their concerns, the situation that triggers the design recovery process, the available resources, and the objectives to be achieved. This phase consists of three sequential activities with their respective milestones: 1) Context Defining: defining the scope of the problem, 2) Feasibility Analysis: determining the feasibility of the reverse engineering process, and 3) Process Planning: establishing the action plan. To specify the scope of the reverse engineering process situation, two goals are set: 1) Define the context, determined by the scope, the stakeholders and their concerns, the process objectives, and the available resources. 2) Describe the situation where the reverse engineering process takes place, for which the causes are established, the circumstances of the context are determined, and the potential consequences are defined, in order to establish the process objective. The result of this activity is the document with the description of the problem context.

The feasibility analysis process determines whether all the required resources are available to achieve the established objective for the reverse engineering process. The goal of this activity is to identify the project's feasibility. To accomplish this, it is recommended to: 1) Define the views that must be recovered based on the context. 2) For each view, identify the required software artifacts. 3) Verify the availability of these artifacts. If they are not available, the process ends. 4) Identify the required techniques and tools for recovering the target views. 5) Check the availability of the required tools. If the tools are not available, the process ends. 6) Determine if it's necessary to qualify the participants or have specialized personnel and define the respective costs. 7) Determine the cost of the reverse engineering process and its viability based on the cost-benefit relationship. The outputs of this activity are the decision on the feasibility of the reverse engineering process, the target views, the required and available artifacts, the cost of the process, and the tools and techniques to be used. Finally, in the inception phase, the sequence of activities for the reverse engineering

doi: 10.25100/iyc.v26i1.12840

process is defined based on the participants' experience, assigning responsibilities and deliverables. The output is the action plan, the resources, and the responsible parties.

The purpose of the data extraction phase is to identify the structural and low-level behavioral elements of the software, along with their relationships. It consists of two activities: 1) Defining the system's domain model using conceptual modeling techniques, and 2) Decomposing the system artifacts to obtain the target views using transformation techniques. Depending on the context in which the reverse engineering process is conducted, the first activity may be performed iteratively and concurrently with activities from the inception and information exploration phases. This is because it helps identify the artifacts to be recovered, guides participants in defining the process feasibility, process plan, and resource allocation. Additionally, it can be used for model abstraction. The inputs for this activity include expert knowledge, problem description, and participant input. In the second activity, static and dynamic analysis techniques are applied to the artifacts constituting the software implementation to identify its elements and relationships. The outcome of this activity is the low-level system model represented in formats such as KDM, FAMIX, Rigi Standard Format, among others (5).

In the knowledge organization phase, the purpose is to transform the low-level models obtained in the previous phase into high-level models, representing and storing them in formats that facilitate their retrieval and analysis. The activities involved are: 1) Model abstraction: Expert domain knowledge is utilized, and mapping rules are applied to transform the extracted artifacts into high-level system elements constituting the target views. This is achieved using manual, semi-automatic, and automatic knowledge organization techniques. 2) Model representation: The goal here is to represent the UML in XMI format of the elements and relationships obtained in the model abstraction, thus achieving the milestone of this phase.

The information exploration phase culminates in the preparation of a report detailing the recovered design and its analysis, arguing the achievement of the established objectives for the reverse engineering process. Therefore, it provides mechanisms for presenting, visualizing, navigating, and analyzing the results of the reverse engineering process. The activities involved are: 1) Visualization: The results obtained in the previous phase are presented graphically to facilitate interpretation and understanding for the process participants. This can be done at the architecture, class, and/or source code level using modeling tools and code editors. This activity applies metaphors and mapping techniques to present the target views and is complemented by navigation strategies using hyperlinks to allow exploration of the recovered views. 2) Results Analysis: The goal is to interpret the recovered views. Inference techniques are applied to analyze the views, with support from domain experts in the application and problem. The proposed query mechanism can be used as a supporting tool in this activity.3) Results Formulation: The obtained results are organized and presented in a final report to facilitate interpretation by the process participants. Depending on the participants' working style, the activities of this phase can be performed sequentially or concurrently.

Query Mechanism

A mechanism was designed and implemented to facilitate the analysis of the recovered views (36, 37) in their KDM and/or UML representations in XMI using the XQuery language. The mechanism employs a text interface that accepts queries in natural language (Spanish) and applies them to the recovered views. It comprises two groups of

doi: 10.25100/iyc.v26i1.12840

elements (see Figure 5). The first group includes artifacts received as input or generated as output, such as: 1) Original Query: The question formulated in natural language. 2) Processed Query: The statement expressed in XQuery. 3) Result: The response in natural language to the original query. 4) UML Model: A file containing the recovered UML views expressed in XMI. 5) KDM Representation: A repository with system elements, their relationships, and operating environments expressed under the KDM specification.

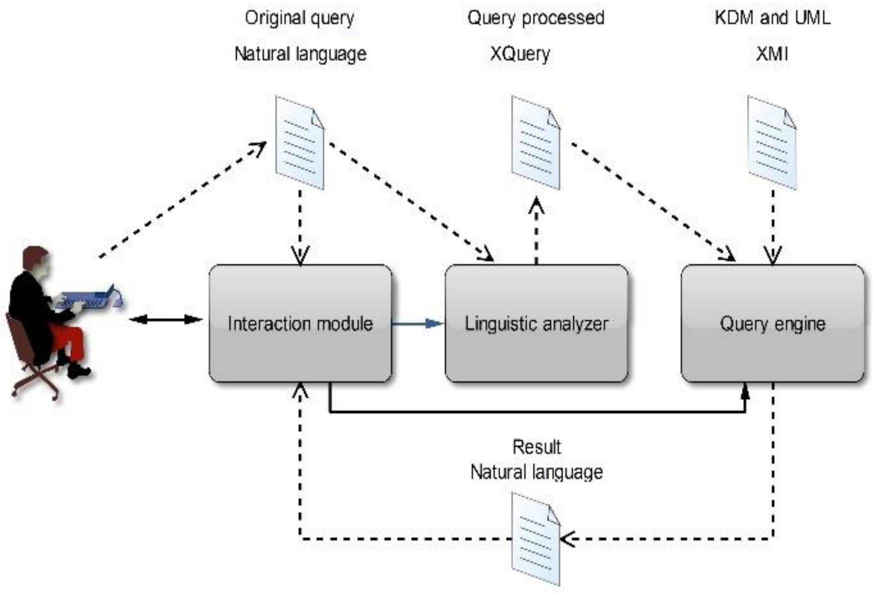


Figure 5. Query mechanism

The second group comprises the components of the mechanism, consisting of: 1) Interaction Module: Allows input of the original query and visualization of the results. 2) Linguistic Analyzer: A finite automaton validates the syntax and grammar of the query. If correct, it transforms the original query into a processed query. 3) Query Engine: Generates the result by executing the processed query on the repository containing the models. The query mechanism operates with the following algorithm: The query is written in natural language in the interaction module, the linguistic analyzer validates the syntax and grammar. If correct, the type of query (simple query, compound query, or metric calculation) is determined. Next, the query engine executes the processed query depending on the identified type, and finally, the interaction module presents the results.

Contexts of use

As a result of the research, the following contexts of reverse engineering use were determined (23): 1) Software Production: In software lifecycle processes such as documentation, maintenance, asset reuse, and verification. 2) Cybersecurity: To define strategies to address risks and issues caused by malicious code that may be present in the system (29-31), facilitating its analysis and understanding. It can also be used to identify potential security vulnerabilities in software systems (25,29). 3) Computer Forensics: To construct evidence that demonstrates facts and allows the formulation of hypotheses (32), facilitating the recovery and presentation of electronically processed data stored in computational media. 4) Education: Used as a didactic tool to facilitate learning based on real cases of successes and failures, stimulate curiosity, and foster the development of design, programming, and maintenance skills (26-28). Additionally, it

doi: 10.25100/iyc.v26i1.12840

facilitates the understanding of concepts in the field of software engineering, such as pattern identification using reverse engineering techniques (22).

Reference Framework evaluation

The analysis of the evaluation results is conducted from two perspectives. Initially, it is based on the outcomes of the case study, and subsequently, on the comparison of the reference framework with other similar approaches. Table 3 presents the results of the case study highlighting the achievement of milestones set for each phase of the methodology proposed in the Reference Framework.

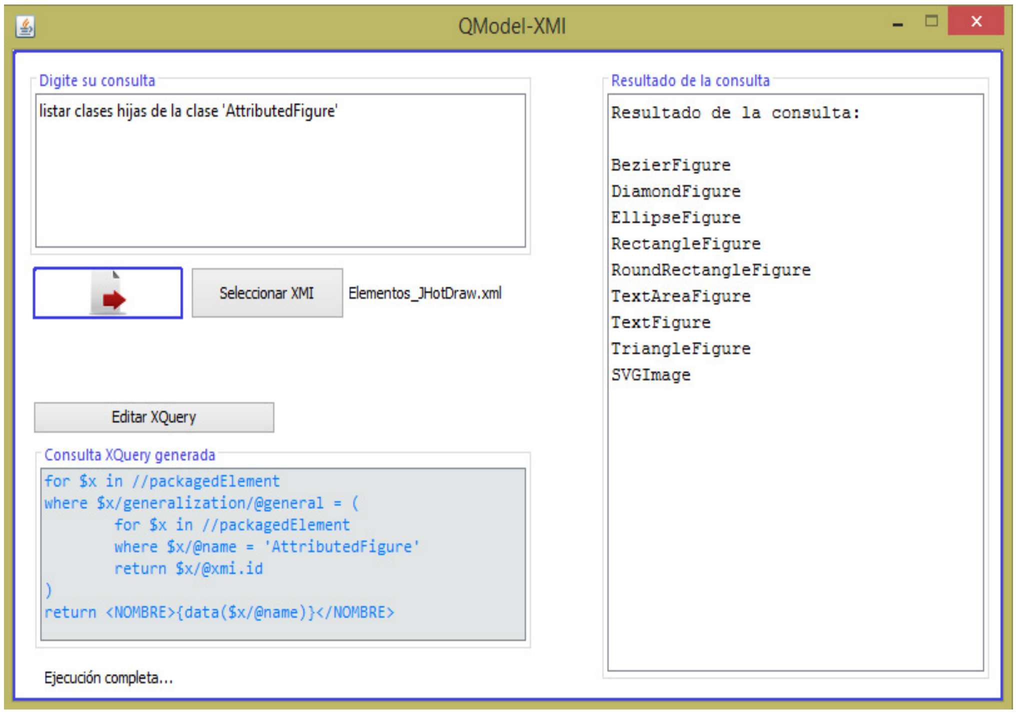
The effectiveness of the Reference Framework was calculated based on Equation (1). The achieved result corresponds to the degree of accomplishment of the objectives set for each unit of analysis. In the educational context, the objective comprises three components: understanding polymorphism concept, identifying its usage in the source code, and its application by the student, as shown in Table 4. In the software production context, the assessment of the degree of achievement of the purpose of each recovered view was made by the software architect, considering the utility that each generated view represented when making decisions to extend the system's functionality, using a scale from one to ten, where 1 indicates that it did not represent any utility and 10 that it was completely useful. In both cases, the effectiveness of the Reference Framework was greater than 80%, thus fulfilling this purpose. Figures 6 and 7.

doi: 10.25100/iyc.v26i1.12840

Table 3. Case study results for each unit of analysis.

Phase		First Unit of Analysis	Second Unit of Analysis	
Inception	Context	Education	Software Production	
	Scope	Teaching - Learning Object Oriented Programming (OOP)	Development of custom software for a company in the industrial sector of Cartagena	
	Purpose	Develop programming skills by applying the OOP paradigm	Extend the functionalities of the CPL System version 1.0	
	Situation	Conducting a course on OOP in the Systems Engineering program of the University of Cartagena	The new software architect does not know the system and needs to add functionalities at the request of the Company's management. There is no system documentation	
	Aim	Understand the concept of polymorphism and develop skills to use it correctly through laboratory practic.	Recover the logical view and system deployment view to extend its functionalities.	
	Stakeholders and concerns	16 students. They want to learn the study topic. A teacher. He wants to guide the learning process.	Product Architect. He must guarantee the architectural integrity of the CPL System 5 Programmers. They must develop the new functionalities of the system based on the architectural decisions made Expert in the problem domain. He must collaborate in the reconstruction of the system information Company Manager. He wants to extend the functionality of the system	
	Resources	The software product under study was JHotDraw version 7.0.6 Enterprise Architect tool: to retrieve artifacts and visualize them Imagix 4D: to calculate source code metrics (10) QModel-XMI: to analyze the recovered models	CPL System version 1.0 Business process models Data dictionary Enterprise Architect tool: to retrieve artifacts and visualize them Imagix 4D: to calculate source code metrics (10) QModel-XMI: to analyze the recovered models	
	Artifacts to be recovered	Modules (Components) of the system System classes	Modules (Components) of the system System classes System Deployment View	
		Recovered artifacts		
	Responsible	Artifac	Responsible	Artifac
Data extraction	Teacher	Domain of the analyzed system: Product developed in Java used by multiple applications to create and edit graphics.	Expert in the problem domain	CPL System domain: plastic packaging production control. It allows inventory control of production and raw materials used, generating performance indicators that support management decisions.
		Using Imagix 4D he identified that JHotDraw has 1,217 classes, 37 interfaces organized in 122 packages, 32,054 code lines and 18,931 comments lines.	Software Architect	Using Imagix 4D he identified that the system is a Java application, made up of 1,302 classes and 3 interfaces distributed in 137 packages. It has 66,095 code lines and 9,872 comments lines. For persistence use MARIADB together with SQL Server interfaces
	Students	Classes model in XMI format: They used Enterprise Architect to recovery this view from JHotDraw.	Developers	System class model in XMI format: They used Enterprise Architect to recovery the class model of each system package.
Knowledge organization	Teacher	View of components and connectors: He applied mapping rules and used semi-automatic and manual techniques to obtain a high-level system representation.	Software Architect	View of components and connectors: He applied mapping rules and used semi-automatic and manual techniques to obtain a high-level system representation.
				Deployment View: He analyzed the system infrastructure at runtime.
Information exploration	Students	Lab Report: Used Enterprise Architect to visualize the retrieved views. With the query mechanism they interpreted and analyzed the retrieved views (see Figure 6), which allowed them to identify the application of polymorphism.	Software Architect	With the query mechanism, he analyzed the recovered artifacts and concluded that the product is structured into four layers (See Figure 7). He observed that the system follows good software development practices. He recommended conducting a code cloning analysis to identify reusable assets and debug the system. This information is disclosed without prejudice to the confidentiality agreement signed with the Company
	Teacher	He prepared the report of the academic activity carried out.		

doi: 10.25100/iyc.v26i1.12840



listar clases hijas de la clase 'AttributedFigure'

```
for $x in //packagedElement
where $x/generalization/@general = (
        for $x in //packagedElement
        where $x/@name = 'AttributedFigure'
        return $x/@xmi.id
)
return <NOMBRE>{data($x/@name)}</NOMBRE>
```

Ejecución completa...

```
Resultado de la consulta:

BezierFigure
DiamondFigure
EllipseFigure
RectangleFigure
RoundRectangleFigure
TextAreaFigure
TextFigure
TriangleFigure
SVGImage
```

Figure 6. Use of the query mechanism

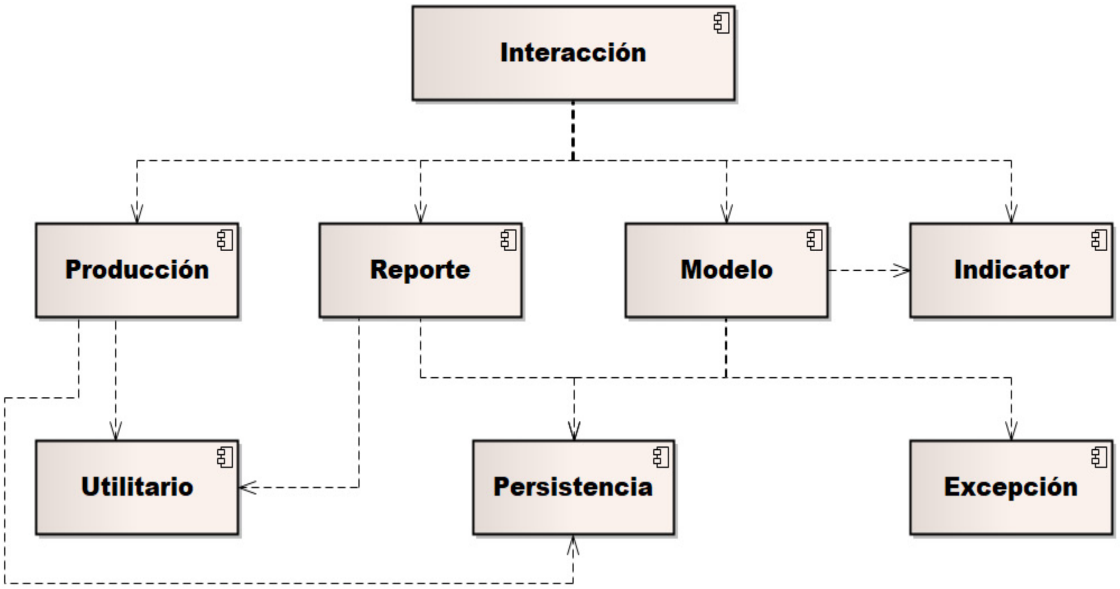


Figure 7. Components and connectors CPL system view.

doi: 10.25100/iyc.v26i1.12840

Table 4. Reference framework effectiveness evaluation

Context	Source	Criterion	RA	RP	effectiveness	
Education	Report on the academic activity conducted by the teacher	The students understood the polymorphism concept	15	16	94%	83%
		The students identified where polymorphism applies	13	16	81%	
		The students applied the polymorphism concept	12	16	75%	
Software production	Software Architect interview	System Modules (Components)	9	10	90%	87%
		System Classes	9	10	90%	
		Deployment view	8	10	80%	

To assess relevance, a survey was conducted with 23 participants in the case study: 16 students, one professor, one software architect, and five developers. The survey aimed to identify their perception using the Lickert scale (38), regarding the degree to which the Reference Framework addresses complexity (HC), considers scope (Sc), purpose (P), situation (S), aims (A), available resources (R), and stakeholders' concerns (C). As shown in Table 5, each of these aspects received an approval rating exceeding 80%. Consequently, it can be affirmed that the Reference Framework effectively guides the software design recovery process in a manner pertinent to the context in which the need arises.

Table 5. Frequency in percentage for the values of the Lickert scale

Cate gories	Scale	HC		Sc		P		S		A		R		C	
		%	n	%	n	%	n	%	n	%	n	%	n	%	n
1	Totally agree	52,17	12	30,43	7	60,87	14	47,83	11	52,17	12	30,43	7	39,13	9
2	Agreed	39,13	9	65,22	15	30,43	7	47,83	11	39,13	9	52,17	12	52,17	12
3	Neither agree nor disagree	4,35	1	0,00	0	8,70	2	0,00	0	8,70	2	8,70	2	8,70	2
4	Disagree	4,35	1	0,00	0	0,00	0	4,35	1	0,00	0	8,70	2	0,00	0
5	Strongly disagree	0,00	0	4,35	1	0,00	0	0,00	0	0,00	0	0,00	0	0,00	0
Total		100,00	23	100,00	23	100,00	23	100,00	23	100,00	23	100,00	23	100,00	23

In the literature review, 112 proposals were identified, of which 51 were selected and classified (see Table 6). Considering that only five define a generic process for software

doi: 10.25100/iyc.v26i1.12840

designs recovery, the comparative analysis also included Modisco (5), Cacophony (17), and Symphony (18) because they are representative references in reverse engineering. All the analyzed proposals belong to the software development domain and are used in maintenance situations (13-18), model-driven reverse engineering (5), and product evolution or migration (12). Similarly, their purpose and participants are specific to the software development context, and only (14) and (15) do not present a conceptual model (CM) to facilitate their interpretation, as observed in Table 7.

Table 6. Proposals classification identified in the literature review

Proposal type	Scopus	IEEE Xplore	ACM Digital Library	Total
Algorithm	3		1	4
Software product analysis techniques	3	1		4
Technique evaluation	3			3
Software maintenance	1			1
Design recovery process	5			5
Systematic review	4	1		5
Technique	23	2	4	29
Total	42	4	5	51

On the other hand, all identified proposals follow the canonical reverse engineering process defined by Tilley et al (19), as observed in Table 8. Therefore, the defined reference framework integrates and complements them by including determinant aspects of the context, allowing the recovery and analysis of the design to be conducted taking into account the stakeholders' concerns, the available resources, and the purposes that arise in different situations in the field of software construction, education, cybersecurity, and forensic computing. It also incorporates a conceptual model that facilitates understanding reverse engineering, which is useful for process participants who are not experts in the field, as observed in the conducted case study. Additionally, the reference framework establishes guidelines and defines a methodology that follows the canonical process (19), including an inception phase to specify the problem context, define the process feasibility, and establish the action plan. Furthermore, it defines a query mechanism implemented in a prototype, which serves to support the analysis of documentation represented in UML models. The main advantage of the query mechanism lies in its ability to ask questions in natural language, making this type of analysis easier for process participants with little experience.

Table 7. Comparison between existing proposals

References	MC	Purpose	Stakeholders
(5)	Yes	Facilitate system design recovery under the MDE approach	Software engineers and developers
(12)	Yes	Recover the architecture of mission-critical systems	Software architects and developers
(13)	Yes	Recover the architecture of product lines	Development team
(14)	No	Recovery the product architecture based on the contextual knowledge for which it was developed	Software engineers, developers, software architects and testing experts
(15)	No	Recover product architecture and evaluate changes	Software engineers and developers
(16)	Yes	Evolution of software architecture	Software engineers and developers
(17)	Yes	Recover system architecture	Developers, integrator, architect, product manager and test architect
(18)	Yes	Recover the necessary views of the software product	Software architects, developers, people responsible for migration or customization of the product, responsible for the problem and commercial representatives

The defined Reference Framework does not incorporate new techniques, nor define new algorithms, nor modify the canonical process of reverse engineering; therefore, it does not rely on specific technologies such as programming languages, application types, among others. It can be used in conjunction with other proposals if circumstances require it. To achieve the process objectives, it is essential to correctly define the scope, resources, stakeholders, and their concerns in each situation that arises, as well as the system views to be recovered. As future work, we propose to define or adapt new design recovery techniques that are intuitive and integrated into the Reference Framework so that they can be used by different stakeholders, regardless of their level of expertise in reverse engineering. Additionally, validating the defined Reference Framework in the contexts of cybersecurity and computer forensics is also proposed.

doi: 10.25100/iyc.v26i1.12840

Table 8. Activities of the canonical reverse engineering process

Activities	References							
	(5)	(12)	(13)	(14)	(15)	(16)	(17)	(18)
Data extraction	Injection	Source code micro analysis, code categorization	Reverse engineering of architectural variants,	Do code analysis		Preparation, extraction	Domain and asset analysis, requirements analysis.	Problem identification, concept
Knowledge			Blocks Identification and functions naming, dependencies identification, construction of multiple views.	Identify standard		Analysis	Specification of metamodels,	Knowledge inference
Information exploration			Derivation of architectural variants	Represent the recovered architecture			Evolution	Information interpretation

Conclusions

The case study results allow us to conclude that the defined reference framework offers a new approach, which guides the software design recovery and documentation retrieved analysis, in a pertinent manner with the characteristics of the stakeholders and their concerns, based on the situations present in the context where it is carried out, taking into account the scope, its purpose, and hiding the complexity of this process from those participants who are not experts in reverse engineering. The reference framework integrates existing approaches (5, 12-18) and complements them by including the context, a conceptual model, a set of guidelines, a methodology, and a query mechanism. It differs from these approaches because it can be used in contexts other than software development. Additionally, in its application, it can be integrated with different techniques and tools.

References

[1] B. Arasteh, R. Sadegi & K. Arasteh. Bölen: Software module clustering method using the combination of shuffled frog leaping and genetic algorithm. Data Technologies and Applications, vol. 55, no. 2,. pp. 251-279. 2021. https://doi.org/10.1108/DTA-08-2019-0138

[2] A. Shatnawi, A.D. Seriai, H. Sahraoui et al. Reverse engineering reusable software components from object-oriented APIs, J. Syst. Softw., 131, pp. 442–460, 2017

[3] F.A. Fontan, M.V. Mäntylä, M.Zanoni et al: Comparing and experimenting machine learning techniques for code smell detection, Empir. Softw. Eng., vol 21, no. 3, pp. 1143–1191 , 2016

[4] J. García, I. Ivkovic, N. Medvidovic. A comparative analysis of software architecture recovery techniques. 28th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE'13), Clayton, Australia, pp. 486–496, 2014

[5] H. Bruneliere, J. Cabot, G. Dupé y F. Madiot, “Modisco: A model driven reverse engineering framework”, Information and Software Technology, vol. 56, no. 8, pp. 1012-1032, 2014. https://doi.org/10.1016/j.infsof.2014.04.007

[6] M. Moser and J. Pichler, “eknows: Platform for Multi-Language Reverse Engineering and Documentation Generation,” 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg, 2021, pp. 559-568, doi: 10.1109/ICSME52107.2021.00057.

[7] T. A. Ghaleb, K. Aljasser and M. A. Alturki, “Enhanced Visualization of Method Invocations by Extending Reverse-engineered Sequence Diagrams,” 2020 Working Conference on Software Visualization (VISSOFT), Adelaide, SA, Australia, 2020, pp. 49-60, doi: 10.1109/VISSOFT51673.2020.00010.

[8] U. Sabir, F. Azam, S. U. Haq, M. W. Anwar, W. H. Butt and A. Amjad, “A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code,” in IEEE Access, vol. 7, pp. 158931-158950, 2019, doi:10.1109/ACCESS.2019.2950884.

[9] Sparx Systems. Architect. User Guide Serieshttps://sparxsystems.com/resources/user-guides/15.2/model-domains/software-models.pdf. 2021

[10] Imagix Corp. Imagix 4D. Disponible en https://www.imagix.com/

[11] C. Stormer, “Software quality attribute analysis by architecture reconstruction (squa3re)”, 11th European Conference on Software Maintenance and Reengineering (CSMR’07), IEEE, 2007, pp. 361-364. https://doi.org/10.1109/csmr.2007.43

[12] D. A. Tamburri y R. Kazman, “General methods for software architecture recovery: a potential approach and its evaluation”. Empirical Software Engineering, vol. 23, no. 3, pp. 1457-1489. 2018. https://doi.org/10.1007/s10664-017-9543-z

[13] M. L. Kerdoudi, T. Ziadi, C. Tibermacine and S. Sadou, “Recovering Software Architecture Product Lines,” 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS), Guangzhou, China, 2019, pp. 226-235, doi: 10.1109/ICECCS.2019.00032.

[14] K. Ibrahim, H. Hassan, K.T. Wassif y S. Makady. Context-Aware Expert for Software Architecture Recovery (CAESAR): An automated approach for recovering software architectures. Journal of King Saud University-Computer and Information Sciences, vol. 35, no. 8, pp. 101-106, 2023.

[15] M. Schmitt Laser, N. Medvidovic, D.M. Le, & J. Garcia. ARCADE: an extensible workbench for architecture recovery, change, and decay evaluation. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering pp. 1546-1550, 2020.

[16] D. Guamán, D., J. Pérez, J. Diaz & C.E. Cuesta. Towards a reference process for software architecture reconstruction. IET Software, vol. 14, no. 6, pp. 592-606, 2020

[17] J. M. Favre, “Cacophony: Metamodel-driven software architecture reconstruction”, 11th Working Conference on Reverse Engineering, IEEE, 2004, pp. 204-213. https://doi.org/10.1109/wcre.2004.15

[18] A. Van Deursen, C. Hofmeister, C, R. Koschke, L. Moonen y C. Riva, “Symphony: View-driven software architecture reconstruction”. Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture, IEEE, 2004, pp. 122-132. https://doi.org/10.1109/wicsa.2004.1310696

doi: 10.25100/iyc.v26i1.12840

[19] S. R. Tilley, P. Santanu y D. B. Smith, “Towards a framework for program understanding”, WPC’96. 4th Workshop on Program Comprehension, IEEE, 1996, pp. 19-28. https://doi.org/10.1109/wpc.1996.501117

[20] G. Granchelli, M. Cardarelli, P. Di Francesco, I. Malavolta, L. Iovino y A. Di Salle, “Towards recovering the software architecture of microservice-based systems”. International Conference on Software Architecture Workshops (ICSAW), IEEE, 2017, pp. 46-53. https://doi.org/10.1109/icsaw.2017.48

[21] M. E. Monroy, J. L. Arciniegas y J. Rodríguez, “Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura”, Información Tecnológica, vol. 27, no. 5, pp. 201-220, 2016. https://doi.org/10.4067/s0718-07642016000500022

[22] H. Zhang and J. Liu, “Research Review of Design Pattern Mining,” 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2020, pp. 339-342, doi: 10.1109/ICSESS49938.2020.9237742.

[23] M. E. Monroy, J. L. Arciniegas y J. Rodríguez, “Caracterización de contextos de uso de la ingeniería inversa”, Información Tecnológica, vol. 28, no. 4, pp. 75-84, 2017. https://doi.org/10.4067/s0718-07642017000400010

[24] IEEE/ISO/IEC International Standard for Software, systems and enterprise--Architecture description, International Organization for Standardization, Ginebra, Suiza, 2022

[25] A. Di Federico, P. Fezzardi and G. Agosta, “rev.ng: A Multi-Architecture Framework for Reverse Engineering and Vulnerability Discovery,” 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Canada, 2018, pp. 1-5, doi: 10.1109/CCST.2018.8585654.

[26] M. E. Monroy, G. E. Chanchí y M. A. Ospina, “Desarrollo de habilidades técnicas en ingeniería de software aplicando ingeniería inversa”, Revista Boletín Redipe, vol. 11, no. 1, pp. 534-550, 2022. https://doi.org/10.36260/rbr.v11i1.1661

[27] E.J. López, M.A. Flores, G.L. Sandoval, B.L., Velázquez, J.J., Vázquez & L.A. Velásquez. Reverse engineering and straightforward design as tools to improve the teaching of mechanical engineering. Industry Integrated Engineering and Computing Education: Advances, Cases, Frameworks, and Toolkits for Implementation, pp. 93-118. 2019.

[28] “I. Verner & M. Greenholts. Teacher education to analyze and design systems through reverse engineering. In Educational Robotics in the Makers Era 1. Springer International Publishing, pp. 122-132, 2017.

[29] A. Sejfia, “A Pilot Study on Architecture and Vulnerabilities: Lessons Learned,” 2019 IEEE/ACM 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE), Montreal, QC, Canada, 2019, pp. 42-47, doi: 10.1109/ECASE.2019.00015.

[30] A. P. David, Ghidra Software Reverse Engineering for Beginners: Analyze, identify, and avoid malicious code and potential threats in your networks and systems, Packt Publishing, 2021.

[31] M. F. Ismael and K. H. Thanoon, “Investigation Malware Analysis Depend on Reverse Engineering Using IDAPro,” 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM), Mosul, Iraq, 2022, pp. 227-231, doi: 10.1109/ICCITM56309.2022.10031698.

[32] K. Hausknecht and S. Gruičić, “Anti-computer forensics,” 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2017, pp. 1233-1240, doi: 10.23919/MIPRO.2017.7973612.

doi: 10.25100/iyc.v26i1.12840

[33] M. D. Peters, C. Marnie, A.C. Tricco, D. Pollock, Z. Munn, L. Alexander, L., ... & H. Khalil, (2020). Updated methodological guidance for the conduct of scoping reviews. JBI evidence synthesis, vol. 18, no. 10, pp. 2119-2126.

[34] R. K. Yin, "Case study research: Design and methods", Sage publications, 2013.

[35] S. L. McGregor y J. A. Murnane, "Paradigm, methodology and method: Intellectual integrity in consumer scholarship", International journal of consumer studies, vol. 34, no. 4, pp. 419-427, 2010. https://doi.org/10.1111/j.1470-6431.2010.00883.x

[36] M. E. Monroy, J. L. Arciniegas y J. C. Rodríguez, "Mecanismo de Consulta para el Análisis de Arquitecturas Recuperadas". Información tecnológica, vol. 28, no. 5, pp. 87-100, 2017. http://dx.doi.org/10.4067/S0718-07642017000500011

[37] M. E. Monroy, J. C. Rodríguez y P. Puello, "QModel-XMI: un mecanismo de consulta para modelos XMI", Revista Espacios, vol. 41, no. 5, pp. 218-228, 2020. https://doi.org/10.48082/espacios-a20v41n45p17

[38] A.T. Jebb, V. Ng, & L. Tay. A review of key Likert scale development advances: 1995–2019. Frontiers in psychology, Vol. 4, no. 12, pp. 637547. 2021.

doi: 10.25100/iyc.v26i1.12840