Using Reverse Engineering in Software Engineering Learning

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

Gabriel E. Chanchí
Systems Engineering Program
Universidad de Cartagena, Cartagena, Bolívar, Colombia
Email- gchanchig@unicartagena.edu.co

Manuel Ospina
Systems Engineering Program
Universidad de Cartagena, Cartagena, Bolívar, Colombia
Email- mospinaa@unicartagena.edu.co

Abstract- Today's environment demands new professional training ways which improve teaching and learning practices adjusted to the characteristics of new generations. We propose a pedagogical strategy based on reverse engineering and problem-based learning (PBL). Action-research methodology was applied in this work. The proposed pedagogical strategy is structured in a set of principles, guidelines, classroom activities and recommendations. The study results allowed us to conclude the pedagogical strategy helped to stimulate student motivation and to contribute to the achievement of learning outcomes, because of the students behavioral, cognitive and technical skills were developed.

Keywords – ***Reverse engineering, software engineering, pedagogical strategy, skills, classroom activities***

I. INTRODUCTION

The future software engineers education is a challenge which demands the students to be active learners, by necessity [1]. Furthermore, as stated by Frand, the today's student is characterized by [2]: 1. Computers aren't technology, 2. Internet is better than TV, 3. Reality is no longer real, 4. Doing is more important than knowing, 5. Learning more closely resembles Nintendo than logic, 6. Multitasking is a way of life, 7. Typing is preferred to handwriting, 8. Staying connected is essential, 9. There is zero tolerance for delays and 10. Consumer and creator are blurring. On the other hand, the Organization for Economic Co-operation and Development (OECD) reported one of the main challenges of the education system is to improve teaching practices at every level [3]. Therefore, it is required to make changes in the teaching-learning process. To achieve this goal, the constructivist approach proposes that the teacher must encourage students' meaningful learning, in order that they can connect their knowledge construction processes to the culturally organized collective knowledge [4].

The Software Engineering Competency Model (SWECOM) [5] includes cognitive skills, behavioral skills and attributes, technical skills, requisite knowledge and related disciplines. Cognitive skills are exhibited in the ability to apply knowledge and reasoning while performing software engineer activities within technical skills areas. Cognitive skills include reasoning, analytical skills, problem solving and innovation. Behavioral attributes are exhibited in the ability to efficiently apply knowledge, cognitive skills, and technical skills. These skills include aptitude, initiative, enthusiasm, work ethic, trustworthiness, willingness, communication skills, cultural sensitivity, team participation skills and technical leadership skills. The technical skills refer to the specialized knowledge and the necessary capacity to do complex actions, tasks, and processes relating to software engineering discipline. In SWECOM, technical skills are grouped as life cycle skill areas (software requirements skills, software design skills, software construction skills, software testing skills and software sustainment skills), and crosscutting skill areas (Software process and life cycle skills, software system engineering skills, software quality skills, software security skills, software safety skills, software configuration management skills, software measurement skills and human-computer interaction skills).

Otherwise, reverse engineering "*is a process of analyzing a subject system to identify the system's components and their interrelationships, and create representations of the system in another form or at higher level of abstraction*" [6], which implies the recovery of implicit knowledge in the system [7]. Reverse engineering is used in multiple disciplines and contexts [8][9]. In mechanical engineering it has been used in the classroom to know and understand products, encouraging innovation skills of students, especially when improving and creating new products [10]. In electronic engineering it has been used to develop cognitive skills such as understanding a product and reviewing physical concepts, and technical skills for example, proposing improvements on the product design, construction and operation [11]. Reverse engineering is currently used in software engineering contexts such as software production, computer security, education, and computer forensics [8][9]. Some studies affirm reverse engineering techniques improve cognitive skills like understanding a system, facilitating the development of technical skills like design and programming [12], contributing to generate enthusiasm in students, and to the systematic development of the analysis capacity and logical thinking [13]. While other studies argue that is necessary to include reverse engineering in the software engineers and professionals training on the computer science field [14][15].

There are proposals based on a socio-constructivist approach and others on a systemic vision perspective, where competencies for software engineers participating in developing and modifying software-intensive systems are defined [5][16]. These proposals define cognitive skills, behavioral skills and attributes, and technical skills, however, the pedagogical perspective absence to achieve the skills development in the classroom is observed as well [14][17][18][19]. Consequently, the main contribution of this paper is the pedagogical strategy based on reverse engineering and problem-based learning. This allows meaningful learnings represented by the skills development, according to the characteristics of today's students. Further, we suggest a set of learning activities to develop behavioral, cognitive and technical skills applying reverse engineering in the classroom.

The rest of the paper is organized as follows. Used methodology is explained in section II. Results and discussion are presented in section III. Concluding remarks are given in section IV.

II. METODOLOGY

The applied methodology was action-research due to the fact that the main objective of this work was to improve teaching practices, instead of generating new knowledge. In addition, "*the thinking here is that research should not only be used to gain a better understanding of the problems which arise in everyday practice, but actually set out to alter things – to do so as part and parcel of the research process rather than tag it on as an afterthought which follows the conclusion of the research*" [20]. For data collection we used participant observations, interviews and repository of students' works, which allowed us to triangulate the information. In action-research the cyclical process comprises five activities [20]: 1) Professional practice, 2) Critical reflection, 3) Research, 4) Strategic planning and 5) Action.

The professional practice in this work includes the teaching practice in the software engineering teaching-learning process at the Faculty of Engineering of University of Cartagena – Colombia. This practice includes the following courses: Algorithms, Programming, Object oriented programming, Data structures, Database, Internet services engineering, Software engineering, Software architecture, Software quality assurance and testing, Software project management and Software engineering advanced topics; which make part of the System Engineering Program curriculum. In the critical reflection phase we made a diagnosis of the situation and identified the problem, therefore the following questions were asked: 1) How is the teaching practice of software engineering being applied?, 2) what would we like to change? and 3) how can we improve this situation?. In the research phase we made a systematic and rigorous inquiry using data collection techniques mentioned above. The obtained results allowed us to define strategic planning that made possible translate the findings into an action plan. Finally, in the action phase we lead to change by putting into practice a set of improvements obtained as a research result, and that are proposed as the main contribution in this paper.

III. RESULTS AND DISCUSSION

In this section, we explain first the pedagogical strategy proposal made. Then, we present a list of classroom activities based on reverse engineering and finally, we analyze the impact that had on our academic program.

By answering the first question (How is the teaching practice of software engineering being applied?), we confirm an eclectic pedagogical model is being used in accordance with institutional guidelines focused on student learning, which makes possible to achieve the learning objectives. Faculty use pedagogical strategies such as project-based learning which contribute to the achievement of learning objectives established in the curriculum. Both, faculty and students recognize the benefits of the teaching practices on the teaching-learning process. Nevertheless, we also identified: 1) it is difficult to spark and keep students' motivation, 2) Faculty believe the development of behavioral, cognitive and technical skills can be further improved, 3) Meaningful learning maintained over time can also be

improved, 4) Students state their characteristics and ways of learning are not taken into account, 5) The undergraduate and graduate new evaluation model in Colombia is focused more on learning outcomes than learning objectives, and 6) The University of Cartagena Engineering Faculty was in the ABET (Accreditation Board for Engineering and Technology) accreditation process.

Therefore, we determined it was necessary to complement two teaching practices aspects: 1) Increasing students' motivation and 2) improving skills development in students ensuring the achievement of learning outcomes. To achieve this purpose, we defined a pedagogical strategy based on reverse engineering and problem-based learning [21]. As shown in Figure 1, we assume the students have their own behavioral attributes and skills (I like to do ...). If a problem about a software product already built is proposed to them, when it's solved they develop cognitive skills, using reverse engineering (I know how …), which improves their learning process becoming future professionals with technical skills (I can do …). The pedagogical strategy is structured in a set of principles, guidelines, classroom activities and recommendations listed below.

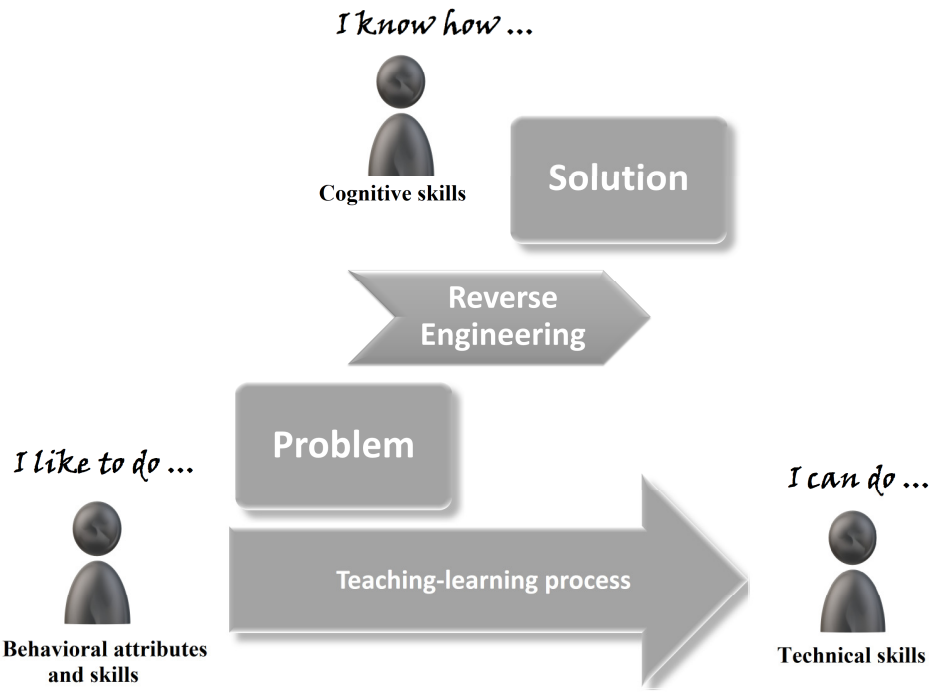

Figure 1. Pedagogical strategy

We assume the following statements as principles to support the pedagogical strategy:

- Learning is a voluntary action. If I want to motivate the students, I must take into account their qualities, specifically [2]: Doing is more important than knowing, learning more closely resembles Nintendo than logic, multitasking is a way of life, staying connected is essential, there is zero tolerance for delays and consumer (student) and creator (teacher) are blurring.
- "*I see and I forget. I hear and I remember. I do and I understand*" (Chinese proverb). Consequently, when the students face a problem, they have a higher probability of learning than when they listen to a lecture.
- When I disassemble a product, I learn to identify its parts and how they work. Reverse engineering is used to analyze and understand a product.
- I learn by identifying good practices in order to repeat them and bad practices in order to avoid them. Reverse engineering provides an opportunity to learn from the successes and mistakes.

In order to achieve better results in classroom activities we defined the following guidelines:

- For each activity, define the learning outcomes based on the development of behavioral, cognitive and technical skills. Table 1 shows some cognitive skills, and Tables 2 and 3 show technical skills [5] related to a set of activities which can be performed using reverse engineering.

Table -1 Cognitive Skills

Skill	Specific skill	Classroom activities using Reverse engineering
Reasoning	Inductive reasoning	✓ Identify more frequently good coding practices ✓ Identify more frequently bad coding practices and correct them ✓ Identify duplicate code
	Deductive reasoning	✓ Identify the use of a specific design pattern ✓ Identify the use of a specific architectural pattern ✓ Identity the bugs in a source code
	Use of abstraction	✓ Manually transform code into a UML model ✓ Identify the components in a software system ✓ Replace duplicate code
	Associative reasoning	✓ Identify duplicate code ✓ Identify the use of a specific design pattern ✓ Identify the use of a specific architectural pattern ✓ Identify the use of the standards in a source code
	Synthesis reasoning	✓ Replace duplicate code
Analytical	Root cause analysis	✓ Identify the bugs root cause in a source code
	Risk identification	✓ Identify possible risks in a source code ✓ Identify the use of coding techniques to minimize threats and bugs propagation
	Impact analysis	✓ Identify the effect of a change in a source code
Problem solving	Divide and conquer	✓ Identify the components in a software system ✓ Identify the use of architectural patterns
	Analogy and reuse	✓ Identify reusable components ✓ Reuse the identified component in a new situation
	Pattern recognition	✓ Identify the use of a specific idiom ✓ Identify the use of a specific design pattern ✓ Identify the use of a specific architectural pattern
	Incremental approaches	✓ Identify the extension point in a software system ✓ Add functionality to the software system
Innovation	Modeling	✓ Add functionality to the software system

Table -2 Technical skills: Life cycle skills areas

Life cycle skill areas	Specific skill	Classroom activities using Reverse engineering
Software Designing skills	Software design fundamental	✓ Identify enabling techniques (such as information hiding, abstraction, coupling/cohesion, and so forth) in a source code. ✓ Identify exception handling and fault tolerance techniques in a source code. ✓ Restructure and refactoring a software system. ✓ Identify the use of design techniques in concurrency areas, event handling, data persistence, or distributed software.
	Software design strategies	✓ Identify used strategies in the software design (such as top-down or bottom-up, stepwise refinement, use of patterns and pattern languages, iterative and incremental processes, and so forth).
	Software Architectural design	✓ Identify the use of architectural styles, views, models, and patterns to specify the high-level organization of a software system. ✓ Identify the component interfaces. ✓ Redesign software components and modules using models, notations, design patterns, and diagramming techniques.
	Software design quality analysis	✓ Do a software design review.
	Software design quality evaluation	✓ Evaluate software design quality
Software construction skills	Detailed design	✓ Redesign a software system by minimizing complexity and enhancing quality. ✓ Identify the appropriate use of design patterns.
	Coding	✓ Identify duplicate code and replace it. ✓ Refactor code when needed. ✓ Identify the use of standards in a source code ✓ Identify the use of defensive coding techniques to minimize bugs and threats propagation.
Sustainment skills	Software maintenance	✓ Obtain and maintain baseline software artifacts. ✓ Perform problem identification and technical impact analysis. ✓ Make changes in a software product (corrective, adaptive, perfective). ✓ Perform preventive maintenance and software re-engineering.

Table -3 Technical skills: Crosscutting skills areas

Crosscutting skill areas	Specific skill	Classroom activities using Reverse engineering
Software Systems engineering skills	System design	✓ Identify system components and specify relationships and interfaces among components. ✓ Conduct trade studies to identify major system components for hardware/software operations.
Software quality skills	Reviews	✓ Collect and analyze appropriate data resulting from a review. ✓ Identify, assign, and perform necessary corrective actions.
	Audits	✓ Collect appropriate data resulting from an audit. ✓ Collect and analyze data collected from an audit. ✓ Establish and implement appropriate solutions for identified problems.
Software Security Skills	Design	✓ Redesign threats and associated risks of modified systems. ✓ Identify the use of design principles to create secure systems. ✓ Identify the use of appropriate, secure design patterns.
	Construction	✓ Identify the use of recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). ✓ Identify the use of recommended coding standards to avoid security vulnerabilities.
	Quality	✓ Perform code reviews to identify security vulnerabilities.
Software Safety Skills	Design	✓ Identify risks from a safety perspective. ✓ Verify completeness and correctness of the design from a safety perspective. ✓ Ensure safety requirements are met.
	Construction	✓ Identify the use of coding standards to assure code safety. ✓ Identify components and their interfaces to avoid safety violations, considering safe coding practices. ✓ Verify the design safety aspects are implemented in a source code.
	Quality	✓ Collect data and report about the safety aspects of the product

- Carefully, identify each of the problems addressed, taking into account the established learning outcomes and the system's source code availability which the students will work, and the necessary reverse engineering tools.
- The problem statement must force students to identify, prevent and/or correct bugs in a source code, or to redesign, modify or improve a software product. We suggest to apply the Diana Wood's recommendations to create effective problem based learning scenarios [21].
- Define the assessment rubric listing the criteria which articulates the expectations for assignments and performance tasks, and describing levels of quality for each criteria [22].
- Always keep in mind the classroom activity must force students to do a reverse engineer process on a source code.
- Apply the problem-based learning process. It is a cyclical process including the following steps [23]: define the problem, generate alternative solutions, evaluate and select a possibility, implement and follow up the solution.
- Guide the students in order to not lose focus on solving the problem. We observed occasionally the students spent more time doing reverse engineering process were very curious about the analyzed product and forgot the classroom activity's main purpose. For this reason, it is necessary for the faculty to guide the students in order to not lose their main objective's attention.

The problems statement was one of the greatest challenges we faced. For this reason, we defined a set of classroom activities which are organized as shown in the Tables 1, 2 and 3. Software engineers cognitive skills include four classifications [5]: Reasoning, analytical, problem solving and innovation skills. Each of these classifications includes specific skills. For instance, reasoning skills include: inductive reasoning, deductive reasoning, heuristic reasoning, use of abstraction, hierarchical and associative reasoning skills. In Table 1, we illustrate – not exhaustively – a set of classroom activities using reverse engineering for each specific skill type. Similarly, software engineer technical skills are grouped as life cycle skill areas and crosscutting skill areas [5]. A life cycle skill area includes skills needed to accomplish various work activities within a development or sustainment software phase. A crosscutting skill area applies through all life cycle skill areas and even in some cases, may apply to

other crosscutting skill areas. In tables 2 and 3, we present – not exhaustively, again – a set of classroom activities using reverse engineering for these skills, respectively.

Additionally, when we began to apply this pedagogical strategy, we also faced challenges such as: the difficulty of finding software as product under study for classroom activities; the lack of knowledge of a methodology to do the reverse engineering process on the education context; the software incorrect selection as product under study in classroom activities; the difficulty when formulating the problems addressed, appropriately in classroom activities; among others. To address these concerns, we suggest the following recommendations:

- Build a software products repository which serves as product under study for classroom activities. This repository must be classified considering criteria, such as: The use of design patterns and/or architectural patterns, the application of good practices, the bugs presence, among others.
- Use ARCo methodology [24][25] when applying the reverse engineering process. It is the only one which the process is implemented, adjusted to the context.
- The software product used in the classroom activity should be carefully selected in order to match the learning outcomes. The incorrect selection of the software product used prevents the achievement of the learning outcomes proposed in the classroom activity.
- For better results, the student must be familiar with the reverse engineering process. We suggest doing reverse engineering previous activities with the students, in order to know the process and tools that can be used.
- The problems addressed should be appropriate to the curriculum level and the student understanding level. It is relevant to promote the cognitive skills development in lower grades, and to encourage technical skills in higher grades.
- Keep in mind that involving reverse engineering as a pedagogical strategy requires a previous work of classroom activities planning and design to achieve the learning outcomes.

This pedagogical strategy has been used in some object oriented programming courses, software engineering courses and software architecture courses. When we analyzed the results, we observed: 1) The student curiosity was stimulated, generating interest in discovering an already developed product. 2) The student satisfaction feeling stimulates the desire to learn when solving the problem. 3) When the student reverse engineer a software product learns replicating good practices and fixing bugs. 4) The learning outcomes were achieved, since the students developed behavioral, cognitive, and technical skills. This represented meaningful learning in the students. 5) The students preferred making video tutorials explaining what was done, rather than writing technical reports.

This research work does not attempt to establish generalizations about the reverse engineering use in a teaching-learning process as a didactic tool. It is only limited to disseminate the results obtained in this research, and to present some recommendations. All of this in a professional training context in the software engineering field. Also, we made clear that only the external context (the classroom) was taken into account for the motivation assessment. The main reason is that the internal context analysis implies the detailed study of psychological and cognitive aspects to all of the students participating in the process since this is beyond the investigation scope.

IV.CONCLUSION

The proposed pedagogical strategy based on the use of reverse engineering and the problem-based learning, stimulated the student motivation, and contributed to the achievement of learning outcomes, since the students behavioral, cognitive, and technical skills were developed. We confirm the development of skills in the classroom is only possible when it is addressed from the pedagogical perspective [14]. Likewise, we confirm the importance of involving reverse engineering in the engineering programs curricular proposals [12][14][15]. The proposed pedagogical strategy is not intended to replace similar strategies; it may be used to complement the teaching-learning process.

REFERENCES

[1] G. C. Gannod, J. E. Burge, and M. T. Helmick, “Using the Inverted Classroom to Teach Software Engineering,” 2008, no. February, pp. 777–786, doi: 10.1145/1368088.1368198.

[2] J. L. Frand, “The Information-Age Mindset: Changes in Students and Implications for Higher Education,” *Educ. Rev.*, vol. 35, no. 5, pp. 15–24, 2000.

[3] OECD, “Educación en Colombia. Aspectos destacados.,” 2016. .

[4] A. F. Díaz and R. G. Hernández, *Constructivismo y aprendizaje significativo. Estrategias docentes para un aprendizaje significativo*. Mexico, 1999.

[5] IEEE, *Software Engineering Competency Model*. IEEE Computer Society, 2014.

[6] E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: A taxonomy," *IEEE Softw.*, vol. 7, no. 1, pp. 13–17, 1990.

[7] P. Tonella, M. Torchiano, B. Du Bois, and A. Systa, "Empirical studies in reverse engineering : state of the art and future trends," *Empir. Softw. Eng.*, vol. 12, no. 5, pp. 551–571, 2007, doi: 10.1007/s10664-007-9037-5.

[8] M. Monroy, J. L. Arciniegas, and J. C. Rodríguez, "Characterization of the contexts of use of reverse engineering," *Inf. tecnológica*, vol. 28, no. 4, pp. 75–84, 2017.

[9] M. Monroy, J. L. Arciniegas, and J. C. Rodríguez, "Modelo Ontológico para Contextos de uso de Herramientas de Ingeniería Inversa," *Inf. tecnológica*, vol. 27, no. 4, pp. 165–174, 2016.

[10] G. Luna, E. Jiménez, L. García, and L. Reyes, "The importance of the research programs of reverse engineering in engineering," in *International Conference on Engineering Education ICEE*, 2010, pp. 1–8.

[11] D. A. Ramos, "Uso de la ingeniería inversa como metodología de enseñanza en la formación para la innovación," in *World Engineering Education Forum*, 2013.

[12] M. R. Ali, "Why Teach Reverse Engineering?," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–4, 2005.

[13] I. Klimek, M. Keltika, and F. Jakab, "Reverse engineering as an education tool in computer science," in *Emerging eLearning Technologies and Applications (ICETA), 9th International Conference on*, 2011, pp. 123–126.

[14] M. Monroy, J. C. Rodríguez, and P. Puello, "Reverse engineering as a teaching tool : A case study in the learning of object-oriented programming," in *17th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation, And Infrastructure for Sustainable Cities and Communities,"* no. July 2019, pp. 1–5, doi: http://dx.doi.org/10.18687/LACCEI2018.1.1.65.

[15] T. Cipresso, "Software reverse engineering education," 2009.

[16] IEEE, *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0)*. IEEE Computer Society, 2016.

[17] A. Díaz-Barriga, "Construcción de programas de estudio en la perspectiva del enfoque de desarrollo de competencias," *Perfiles Educ.*, vol. XXXVI, no. 143, pp. 142–162, 2014.

[18] S. Semerikov, A. Striuk, L. Striuk, M. Striuk, and H. Shalatska, "Sustainability in Software Engineering Education : a case of," vol. 10036, 2020.

[19] A. M. Striuk and S. O. Semerikov, "The Dawn of Software Engineering Education," in *CEUR Workshop Proceedings*, 2019, pp. 35–57.

[20] M. Denscombe, *The good research guide for small-sacle socail research projects*. McGraw-Hill Education, 2014.

[21] D. F. Wood, "Problem based learning," *Bmj 326.7384*, pp. 328–330, 2003.

[22] H. G. Andrade, "Using rubrics to promote thinking and learning," *Educ. Leadersh.*, vol. 57, no. 5, pp. 13–18, 2000.

[23] A. Escribano Gonzales and Á. Del Valle López, "APRENDIZAJE BASADO EN PROBLEMAS una propuesta metodológica en Educación Superior," *Rev. Educ. y Desarro. Soc.*, vol. 11, no. 1, pp. 8–23, 2008, [Online]. Available: http://www.untumbes.edu.pe/vcs/biblioteca/document/varioslibros/0296. El aprendizaje basado en problemas. Una propuesta metodológica en educación superior.pdf.

[24] M. Monroy, J. R. Rodríguez, and P. Puello, "A Methodological Approach for Software Architecture Recovery," *Indian J. Sci. Technol.*, vol. 11, no. 21, pp. 1–8, 2018, doi: 10.17485/ijst/2018/v11i21/124487.

[25] M. Monroy, M. Pinzger, and J. L. Arciniegas, "ARCo : Architecture Recovery in Context," *J. Xi'an Univ. Archit. Technol.*, vol. XIII, no. 2, pp. 128–143, 2021.