A Systematic Mapping Study on Software Architecture Recovery

Conference Paper · October 2016

CITATIONS
0

READS
28

4 authors, including:

Gözde Karataş
T.C. Istanbul Kultur University
8 PUBLICATIONS **1** CITATION
SEE PROFILE

Cagatay Catal
Wageningen University & Research
54 PUBLICATIONS **992** CITATIONS
SEE PROFILE

A Systematic Mapping Study on Software Architecture Recovery

Çağatay Çatal[1], Akhan Akbulut[1], Gözde Karataş[2], Zeynel Abidin Sezer[1]
[1]Department of Computer Engineering
[2]Department of Mathematics and Computer Science
Istanbul Kültür University
Bakirkoy, 34156, Istanbul, Turkey
{c.catal, a.akbulut, g.karatas, z.sezer}@iku.edu.tr

Abstract— **In this study, we investigated the approaches used in software architecture recovery papers, identify the current status of paper distributions in terms of year, publication channel, electronic databases, and journals. We executed a mapping study to cluster the software architecture recovery research papers. Papers published since 2000 have been used for this study. The following databases were investigated: Wiley, IEEE, ACM, Science Direct. Our search accessed 250 papers, but after in-depth analysis, 60 papers were found to be related to the software architecture recovery area. Our study shows that there exist many architecture recovery approaches in the literature, with machine learning-based techniques dominating the field. On the basis of this study, we suggest researchers develop more model centric software architecture recovery approaches because of model driven development's popularity in software engineering field.**

Keywords-software architecture; architecture recovery; architectural erosion; architectural drift

I. INTRODUCTION

There are many definitions for software architecture. According to Perry and Wolf [1], *"architecture is a set of architectural elements that have a particular form".* According to Shaw and Garlan [2], software architecture consists of the elements, interactions among the elements, patterns, and constraints on these patterns. According to ANSI/IEEE Standard 1471 [3], *"architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution"*. According to a recent definition of Taylor et al. [4], software architecture is the set of principle design decisions about a software system. We use Taylor et al.'s software architecture definition throughout this paper. This recent definition emphasizes that architecture is updated during the software development life cycle and it is not limited to the design phase. In practice, software companies mostly prefer explaining architecture and the detailed design in the same document known as Software Design Description (SDD). However, it is possible to prepare a separate document called High-Level Design Document, Preliminary Design Document or Software Architecture Document. No matter how it is called, we must ensure that the principle design decisions are documented in this document. During the maintenance phase, a documented architecture simplifies all the steps which are necessary for maintenance. Therefore, architecture and the document which describes it must be updated when a new principle design decision is identified or updated.

Before the implementation phase of the software development, an architecture that includes design decisions must be built. This architecture is called "prescriptive architecture" or "as-conceived" or "as-intended" architecture. After the software system has been implemented, the architecture of this software is called "descriptive architecture" or "as-implemented" or "as-realized" architecture [4]. Current practices show that descriptive architecture deviates from the prescriptive architecture during the software development life cycle. In theory, prescriptive architecture must be updated first during the software evolution, but in practice the descriptive architecture is directly modified and prescriptive architecture is not updated. There are several reasons why this problem occurs. For example, developer may be reluctant to update the prescriptive architecture or short deadlines may prevent this process. Another reason might be related to the missing prescriptive architecture. If prescriptive architecture and descriptive architecture are not consistent, architectural degradation occurs. There are two concepts related to the architectural degradation [4]:

1. *Architectural drift:* If the descriptive architecture includes principle design decisions which do not exist in the prescriptive architecture but which do not violate principal design decisions in prescriptive architecture, this concept is called architectural drift.

2. *Architectural erosion:* If the descriptive architecture includes principal design decisions which violate principal design decisions in prescriptive architecture, this concept is called architectural erosion.

When the architectural degradation occurs, someone should recover the architecture from its implementation-level artifacts such as exe files or .class files. This recovery process is called architectural recovery. Since it's very difficult to recover architecture from such kind of artifacts, descriptive and prescriptive architecture must be synchronously updated [4]. In addition to the "recovery" term, some of the researchers prefer the terms "discovery", "reconstruction", "reconciliation", and "restoration". In this study, a Systematic Mapping Study was performed to cluster the research papers in which approaches are examined in software architecture recovery area. The main motivation of this study was to get an up-to-date overview of the software architecture recovery subject before we start the development of a new software architecture recovery model. After the in-depth analysis of the current literature on this topic, we decided to build a novel recovery technique. Therefore, this study was crucial for further steps of our project. The following approaches have been used to categorize the papers. The first 12 of these methods were from the paper of Ghulam and Nadım [5].

1. Data Flow based
2. Knowledge based
3. Design pattern based
4. Program slicing based
5. Formal method based
6. Program comprehension based
7. Domain based
8. Concept analysis based
9. Machine learning based
10. Metrics based
11. Structural based
12. Evidence based
13. Top-Down
14. Ontology-driven
15. Facet based
16. Inspection based
17. Model centric
18. Bottom-Up

The next sections are organized as follows: Section II describes a detailed process for systematic maps. Section III explains the results and section IV presents the conclusions.

II. METHODOLOGY

Evidence-based Software Engineering (EBSE) approach proposed by Barbara Kitchenham is widely used since 2004 in Software Engineering community. Systematic Literature Reviews (SLR) and Systematic Mapping Studies have been published so far on several software engineering topics such as test case prioritization, software fault prediction, architecture optimization, and global software engineering. For a SLR study, we address the research questions defined in the research paper. However, we try to access all the papers on a specific topic when we perform a systematic mapping study. Therefore, there is a difference between these two different methods used in EBSE. We accessed to software architecture recovery papers from electronic databases such as IEEExplore, ACM, Science Direct, and Wiley. We conducted this study on software architecture recovery topic, reached to architecture recovery papers, and analyzed papers published since year 2000. Since there is no other similar study on this topic, this is the first known mapping study. Planning, execution, and reporting phases are applied in the ascertain of mapping study. Research questions, search criteria, and inclusion/exclusion criteria were identified during the planning phase. The following research questions were defined for this research study:

- RQ1: What are the most investigated software architecture recovery approaches?
- RQ2: Which journals include papers on software architecture recovery?
- RQ3: Which journal is the most dominant journal on software architecture recovery problem?
- RQ4: How is the distribution of papers in terms of year, publication channel, and electronic databases?

We used the following keywords for this study:

- Software architecture recovery approach
- Software architecture discovery approach
- Software architecture reconstruction approach
- Software reverse architecting approach

The following databases were investigated: Wiley, IEEE, *ACM*, and *Science Direct*. This was not a manual search which consumes so much time. We executed the search keywords in databases automatically. Papers published at journals, and conferences were included and studies that are not related to the architecture recovery were not used for the study. The inclusion criteria and exclusion criteria are listed as follows:

- IC1: Primary studies on the architecture recovery
- IC2: Secondary studies on the architecture recovery

- IC3: Studies that compare recovery approaches
- EC1: Studies not addressing architecture recovery
- EC2: Studies written non-English languages
- EC3: Papers without full text access

During the execution step, we searched on databases with the search strings explained previously in this phase. During the reporting step, statistical data were interpreted with related research questions.

III. RESULTS

Figure 1 plots the publication year on the x-axis and depicts the number of papers published in that year on the y-axis. The year 2012 is the peak year on architecture recovery. Although we expected to see more papers within the last three years, this was not the case we've seen from the Figure 1. However, we see that there are still some researchers who try to publish papers on this topic and the topic is still popular in software engineering community. 250 papers were accessed from databases, but we excluded 190 of them since they were not related with the research area. Hence, we used 60 research papers [6-65] to complete our review. Distribution of studies for each database is illustrated in Figure 2.

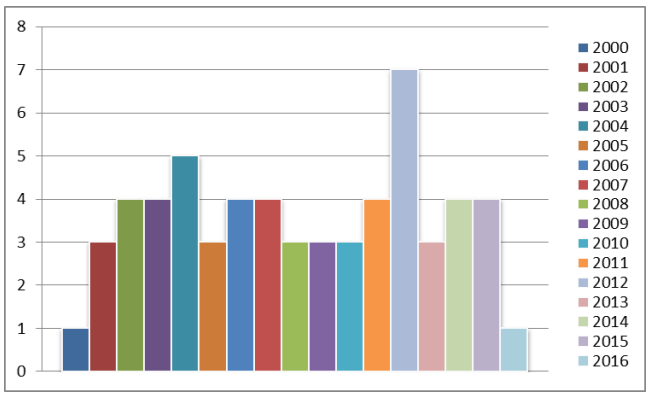

Figure 1. Number of papers per year

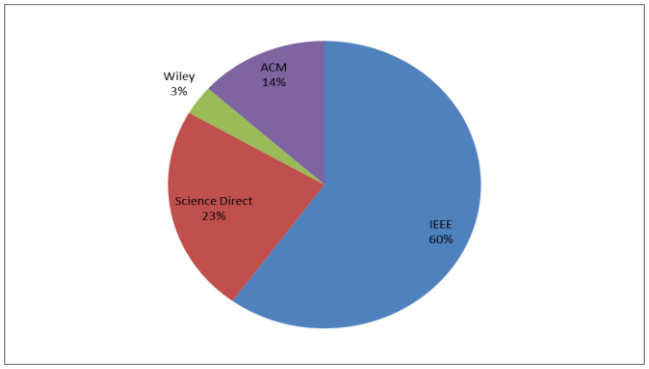

Figure 2. Distribution of studies in each database

Journal Name	# of Papers
Information and Software Technology	2
Information Sciences	3
Journal of Software: Evolution and Process	1
Journal of Systems and Software	5
Science of Computer Programming	2
IEEE Transactions on Software Eng.	2
Performance Validation S.S.	1
Software: Practice and Experience	1
Applied Computing Review	2
Total	**17**

Table 1. # of studies per journal

Only seven-teen of papers were published in journals, whereas 68% were not. Table 1 shows distribution of papers to each journal. According to this table, The Journal of Systems and Software (JSS) is the most popular journal on architectural recovery topic.

Table 2 shows the category distribution of the papers. If a paper fits into more than one category, then we show that paper in those categories instead of selecting one of them. Mostly machine learning based approaches were dominant in our analysis.

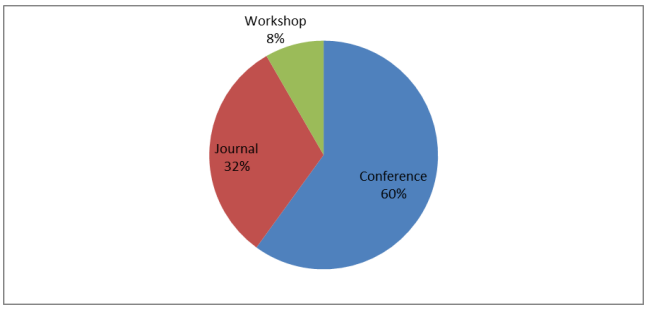

Figure 3. Distribution of publication channel

IV. CONCLUSION AND FUTURE WORK

In this study, we focused on software architecture recovery problem. We performed a systematic mapping study on software architecture recovery. Four electronics databases were used; 250 studies were identified. After inclusion and exclusion criteria were applied, 60 papers were chosen for the mapping study. We observed that classifying software architecture recovery approaches is not an easy task because approaches are not totally isolated. Most of the papers used machine learning-based approaches and there were only three papers which used model centric approaches. Because model driven development is widely used in industry in these days, we suggest researchers to focus on model centric software architecture recovery approaches.

As a future work, we'll perform a systematic literature review on software architecture recovery and we'd like to see the benefits of these approaches based on experimental data if any. Also, the number of papers which apply model-based approaches is not at an acceptable level compared to the other approaches. We will focus on model-based approaches to develop a fully automatic software architecture recovery technique. This systematic mapping study was the first step before we start to design and implement a model-based architecture recovery technique. In addition, we want to use the power of machine learning-based techniques while we design the model-based recovery approach.

Approach	# of Papers
Program slicing based approaches	10
Design pattern based approaches	11
Knowledge based approaches	7
Data Flow based approaches	5
Structural Based approaches	9
Top-Down Approaches	6
Program comprehension based	7
Concept analysis approaches	3
Model Centric Approaches	5
Formal method based approaches	2
Domain based approaches	3
Metrics Based approaches	3
Bottom-Up Approaches	2
Machine Learning-based approaches	15
Evidence Based Approaches	2
Ontology-driven Approaches	1
Facet Based Approaches	1
Graph Pattern Matching Approach	3
Semi-automated Approach	1
Graph Pattern Matching Approach	2
Characterization Approach	1
Semi-automated Approaches	2
Search Based Approaches	1

Table 2. # of studies per category

REFERENCES

[1] D.E. Perry, and A.L. Wolf, "Foundations for the Study of Software Architecture",ACM SIGSOFT Software Engineering Notes, 17, 4, 1992.

[2] D. Garlan, and M. Shaw, "An Introduction to Software Architecture", volume I. World Scientific Publishing Company, 1993.

[3] http://standards.ieee.org/findstds/standard/1471-2000.html

[4] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, "Software Architecture: Foundations, Theory, and. Practice". Wiley, 2009.

[5] R. Ghulam, and A. Nadım, "Software Architecture Recovery", in World Academy of Science, Engineering and Technology 4, pp. 421-426, 2007

[6] Cuadrado, F. ; Garcia, B. ; Dueas, J.C. ; Parada, H.A.. 2008. A Case Study on Software Evolution towards Service-Oriented Architecture. Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on Digital Object Identifier: 10.1109/WAINA.2008.296

[7] Ros, J.P. ; Sangwan, R.S.. 2011. A Method for Evidence-Based Architecture Discovery. Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on Digital Object Identifier: 10.1109/WICSA.2011.54

[8] Trosky B. Callo Arias, Pierre America and Paris Avgeriou. 2013. A top-down approach to construct execution views of a large software-intensive system. Journal of Software: Evolution and Process

[9] Trosky B. Callo Arias, Paris Avgeriou, Pierre America, Krelis Blom, Sergiy Bachynskyy. 2011. A top-down strategy to reverse architecting execution views for a large and complex software-intensive system: An experience report . Science of Computer Programming

[10] Sartipi, K. ; Dezhkam, N. . 2007. An Amalgamated Dynamic and Static Architecture Reconstruction Framework to Control Component Interactions 259. Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on Digital Object Identifier: 10.1109/WCRE.2007.10

[11] Sajnani, H. . 2012. Automatic software architecture recovery: A machine learning approach . Program Comprehension (ICPC), 2012 IEEE 20th International Conference on Digital Object Identifier: 10.1109/ICPC.2012.6240501 "

[12] Claudia López, Víctor Codocedo, Hernán Astudillo, Luiz Marcio Cysneiros. 2012. Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. Science of Computer Programming

[13] Lakshitha de Silva, Dharini Balasubramaniam. 2012. Controlling software architecture erosion: A survey. Journal of Systems and Software

[14] Anton Jansen, Jan Bosch, Paris Avgeriou. 2008. Documenting after the fact: Recovering architectural design decisions . Journal of Systems and Software

[15] Chun Yong Chong, Sai Peck Lee, Teck Chaw Ling. 2013. Efficient software clustering technique using an adaptive and preventive dendrogram cutting approach. Information and Software Technology

[16] Astudillo, H. ; Valdes, G. ; Becerra, C. . 2012. Empirical Measurement of Automated Recovery of Design Decisions and Structure . Andean Region International Conference (ANDESCON), 2012 VI Digital Object Identifier: 10.1109/Andescon.2012.33

[17] Garcia, J. ; Popescu, D. ; Mattmann, C. ; Medvidovic, N. ; Yuanfang Cai . 2011. Enhancing architectural recovery using concerns . Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on Digital Object Identifier: 10.1109/ASE.2011.6100123

[18] Aline Vasconcelos, Cláudia Werner. 2011. Evaluating reuse and program understanding in ArchMine architecture recovery approach . Information Sciences

[19] Chardigny, S. ; Seriai, A. ; Oussalah, M. ; Tamzalit, D. . 2008. Extraction of Component-Based Architecture from Object-Oriented Systems . Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on Digital Object Identifier: 10.1109/WICSA.2008.44

[20] Maqbool, O. ; Babri, H.A. 2007. Hierarchical Clustering for Software Architecture Recovery. Software Engineering, IEEE Transactions

[21] Christoph Stoermer, Anthony Rowe, Liam O'Brien and Chris Verhoef. 2006. Model-centric software architecture reconstruction. Software: Practice and Experience

[22] Lungu, M. ; Lanza, M. ; Girba, T. . 2006. Package patterns for visual architecture recovery . Software Maintenance and Reengineering, 2006. CSMR 2006.

[23] Pirzadeh, H. ; Alawneh, L. ; Hamou-Lhadj, A. . 2009. Quality of the Source Code for Design and Architecture Recovery Techniques: Utilities are the Problem . Quality Software, 2009. QSIC '09. 9th International Conference on Digital Object Identifier: 10.1109/QSIC.2009.69

[24] Chardigny, S. ; Seriai, A. ; Tamzalit, D. ; Oussalah, M. . 2008. Quality-Driven Extraction of a Component-based Architecture from an Object-Oriented System . Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on Digital Object Identifier: 10.1109/CSMR.2008.4493324

[25] Sam Chung ; Daehee Won ; Baeg, M.H. ; Sangdeok Park . 2009. Service-oriented reverse reengineering: 5W1H model-driven re-documentation and candidate services identification . Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on Digital Object Identifier: 10.1109/SOCA.2009.5410445

[26] Ducasse, S. ; Pollet, D. . 2009. Software Architecture Reconstruction: A Process-Oriented Taxonomy . Software Engineering, IEEE Transactions on Volume: 35 , Issue: 4 Digital Object Identifier: 10.1109/TSE.2009.19

[27] Pashov, I. ; Riebisch, Matthias . 2004. Using feature modeling for program comprehension and software architecture recovery. Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the Digital Object Identifier

[28] Sora, I. ; Glodean, G. ; Gligor, M. . 2010. Software Architecture Reconstruction: An Approach Based on Combining Graph Clustering and Partitioning. Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on Digital Object Identifier: 10.1109/ICCCYB.2010.5491289 "

[29] Pollet, D. ; Ducasse, S. ; Poyet, L. ; Alloui, I. ; Cimpan, S. ; Verjus, H. . 2007. Towards A Process-Oriented Software Architecture Reconstruction Taxonomy. Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on Digital Object Identifier: 10.1109/CSMR.2007.50

[30] Solms F. A Systematic Method for Architecture Recovery. 2012.

[31] Ramírez A, Romero JR, Ventura S. An approach for the evolutionary discovery of software architectures. 2015;305:234-255.

[32] Avritzer AA, Bondi AB. Developing Software Reliability Models in the Architecture Phase of the Software Lifecycle. 2014:22-23. doi:10.1109/ISSREW.2014.110.

[33] Chung-Horng Lung, Marzia Zaman, Amit Nandi. 2004. Applications of clustering techniques to software partitioning, recovery and restructuring . Journal of Systems and Software

[34] Li Qingshan ; Chu Hua ; Hu Shengming ; Chen Ping ; Zhao Yun . 2005. Architecture recovery and abstraction from the perspective of processes . Reverse Engineering, 12th Working Conference on Digital Object Identifier: 10.1109/WCRE.2005.6

[35] Bauer, M. ; Trifu, M. . 2004. Architecture-aware adaptive clustering of OO systems . Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on Digital Object Identifier: 10.1109/CSMR.2004.1281401

[36] Favre, Jean-Marie . 2004. CaCOphoNy: metamodel-driven software architecture reconstruction . Reverse Engineering, 2004. Proceedings. 11th Working Conference on Digital Object Identifier: 10.1109/WCRE.2004.15

[37] Maqbool, O.; Babri, H.A. 2005. Interpreting clustering results through cluster labeling. Emerging Technologies, 2005. Proceedings of the IEEE Symposium on Digital Object Identifier: 10.1109/ICET.2005.1558920

[38] Haqqie, S. ; Shahid, A.A. . 2005. Mining Design Patterns for Architecture Reconstruction using an Expert System . 9th International Multitopic Conference, IEEE INMIC 2005 Digital Object Identifier: 10.1109/INMIC.2005.334464

[39] Van Deursen, A. ; Riva, C. . 2004. Software architecture reconstruction . Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on Digital Object Identifier: 10.1109/ICSE.2004.1317516

[40] Pashov, I. ; Riebisch, Matthias . 2004. Using feature modeling for program comprehension and software architecture recovery. Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the Digital Object Identifier

[41] Van Deursen, A. ; Hofmeister, Christine ; Koschke, R. ; Moonen, L. ; Riva, C. . 2004. Symphony: View-Driven Software Architecture Reconstruction. Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on Digital Object Identifier: 10.1109/WICSA.2004.1310696 "

[42] A. E. Hassan and R. C. Holt, “Architecture Recovery of Web Applications,” 2002.

[43] B. Qiao, H. Yang, and W. C. Chu, “Bridging Legacy Systems to Model Driven Architecture,” 2003.

[44] M. Pinzger and H. Gall, “Pattern-Supported Architecture Recovery £,” 2002.

[45] K. Sartipi, “On Modeling Software Architecture Recovery as Graph Matching,” 2003.

[46] I. Ivkovic, “Enhancing Domain-Specific Software Architecture Recovery,” 2003.

[47] K. Sartipi, “A Graph Pattern Matching Approach to Software Architecture Recovery £.”

[48] L. Ding, “Focus : A Light-Weight , Incremental Approach to Software Architecture R-ecovery and Evolution,” 2001.

[49] K. Goševa-popstojanova and K. S. Trivedi, “Architecture-based approach to reliability assessment of software systems,” vol. 45, pp. 179–204, 2001.

[50] J. Lundberg, “Architecture Recovery by Semi-Automatic Component Identification,” vol. 82, no. 5, pp. 98–114, 2003.

[51] Martin Monroy, Member, IEEE, Jose Luis Arciniegas, Member, IEEE, and Julio Rodríguez. 2015. An approach to recovery and analysis of architectural behavioral views. 2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)

[52] L. De Silva and D. Balasubramaniam, “The Journal of Systems and Software Controlling software architecture erosion : A survey,” vol. 85, pp. 132–151, 2012.

[53] R. Roeller, P. Lago, and H. Van Vliet, “Recovering architectural assumptions,” vol. 79, pp. 552–573, 2006.

[54] A. Bere, “Toward assessing the impact of mobile security issues in pedagogical delivery: A mobile learning case study,” Sci. Inf. Conf. (SAI), 2013, pp. 363–368, 2013

[55] A. B. Belle, G. El Boussaidi, and S. Kpodjedo, “Combining lexical and structural information to reconstruct software layers,” vol. 74, pp. 1–16, 2016

[56] G. Scanniello, A. D. Amico, C. D. Amico, and T. D. Amico, “An Approach for Architectural Layer Recovery,” pp. 2198–2202.

[57] N. Ali and J. Buckley, “Characterizing Real-Time Reflexion-based Architecture Recovery : An In-vivo Multi-Case Study,” pp. 23–32, 2012.

[58] F. Solms, “Experiences with using the Systematic Method for Architecture Recovery (SyMAR) Categories and Subject Descriptors,” pp. 170–178.

[59] M. M. R, U. Cauca, and J. R. R, “Marco de referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento Framework for recovery and analysis of behavioral architectural views,” pp. 430–433, 2012.

[60] Y. Cai, H. Wang, S. Wong, and L. Wang, "Leveraging Design Rules to Improve Software Architecture Recovery Categories and Subject Descriptors," pp. 133–142.

[61] A. Van Deursen and P. O. Box, "Software Architecture Recovery and Modelling," pp. 4–7, 2001.

[62] K. Jeet, "Software Architecture Recovery using Genetic Black Hole Algorithm," vol. 40, no. 1, pp. 1–5, 2015.

[63] T. Lutellier, D. Chollak, J. Garcia, D. Rayside, and N. Medvidovi, "Comparing Software Architecture Recovery Techniques Using Accurate Dependencies," 2015

[64] L. Xiao, Y. Cai, and R. Kazman, "Design Rule Spaces : A New Form of Architecture Insight," pp. 967–977.

[65] A. Dragomir, H. Lichter, J. Dohmen, and H. Chen, "Run-time Monitoring-based Evaluation and Communication Integrity Validation of Software Architectures," 2014.