Software Architecture Reconstruction Method, a Survey

Zainab Nayyar
Department of computer Engineering EME, National University of Science and Technology (NUST), H-12, Islamabad, Pakistan

Nazish Rafique
Department of Computer Engineering, College of EME, National University of Science and Technology (NUST),H-12,Islamabad, Pakistan

***Abstract*—Architecture reconstruction belongs to a reverse engineering process, in which we move from code to architecture level for reconstructing architecture. Software architectures are the blue prints of projects which depict the external overview of the software system. Mostly maintenance and testing cause the software to deviate from its original architecture, because sometimes for enhancing the functionality of a system the software deviates from its documented specifications, some new modules are included in the system without modifying the architecture of a system which create issues while reconstructing the system, as much as the software is closed to the architecture the more it is easy to maintain and change the document so the conformance of architecture with the product is checked by applying the reverse engineering method. Another reason for reconstructing the architecture is observed in the case of legacy systems, when they need modification or an enhanced version of the system is needed to be developed. This paper includes the methods and tools involved in reconstructing the architecture and by comparing them the best method for reconstructing architecture will be suggested.**

Keywords—Software architecture; reverse engineering; architecture reconstruction; architecture erosion; architecture mismatch; architecture chasm; architecture drift; forward engineering; architectural aging

I. INTRODUCTION

Many organizations use old softwares but as the new advancements in technology occurring day by day there is often a need to mold the softwares according to the current and latest technological aspects. But sometimes it is difficult to made changes to the code because as the time passes the documents which comprises the implementation of software are outdated or missing. Mostly the idea of developing the new software from scratch is not favored so software architecture reconstruction is used to recover the architecture and then documenting and updating the architecture.

There are certain problems which arise while maintaining and understanding the system. The first problem is that mostly architecture of a system does not explicitly shown in the system unlike classes and packages; another problem is that many large and important applications were developed over time so their architecture drifts [1]. These problems are solved by doing software architecture reconstruction [2].

Software architecture represents the model of the software system which expresses the high level of abstraction. The architectural view of the system hides details of implementation, data representation and algorithms and only concentrates on developing a link between requirements and implementation. The software architecture depicts actually the tangible entities of a system and relationship between those entities. The role of software architecture in developing the software is to understand, reusability, construction, evolution, analysis and management of a system [3].

However, only few organizations participate in software architecture reconstruction efforts. Architecture of software systems plays a significant role in attaining specific business goals. Therefore it is very important to understand the environment of organization and the importance of software architecture so that it is easy to out line the software architecture efficiently [4].

The architecture of software is designed to validate and verify, which requirements can be implemented and which cannot. Architecture of a software system generally restrict the developer within the scope, more the software is closest to the architecture more it is easy to validate its conformance with the requirements.

Architecture reconstruction process is an iterative and interactive approach. It consists of four steps. In the first step, set of views are extracted from software implementation such as source code and dynamic information. These views represent the system's essential structural and behavioral components. Second step consists of fusion of extracted views. It is used to create fused views that enhance and improve the extracted views. In the Third step, the job of analyst is to iteratively and interactively improve and applies design patterns to the fused views to reconstruct the architectural-level views. Design pattern helps analyst to understand the architecture of the system as structural and behavioral relationships among different components. In the last step, derived views are further investigated to evaluate conformance of architecture, to identify goals for reengineering or reuse, and to analyze the essential qualities of architecture [5].

II. LITERATURE REVIEW

This section presents an extended review of the research work that has been done so far regarding the software architecture reconstruction. It also includes the detail discussion on tools and techniques used for reconstruction of architecture.

Software architecture reconstruction terminology is incomplete without including some terms like forward engineering, reverse engineering, architectural aging, architectural drift, architectural erosion and architectural mismatch. Forward engineering includes the normal set of steps required for developing a system in which the system has started from requirement gathering to implementation phase. Where as in reverse engineering the inverse process is carried out in which the programming details are used to get the hidden details about the architecture of the system [7]. For reverse engineering the important data of the system is extracted, than the extracted information leads to the high level design of the system and than the high level information helps the developer to get into the architecture of the particular system [6]. Factors due to which software loose its architecture that leads to an architecture reconstruction are discussed in architecture aging it occurs due to architecture erosion, drift in the architecture or any mismatch occurs in architecture. Mostly the violations of the architecture cause the architecture to erode this scenario is also observed in architecture drift because of the several ambiguities and not developing the system by following the architecture. Sometimes a gap is created between the architecture and code of the system due to maintenance, testing etc. this is known as architecture mismatch [7]. Figure 1 demonstrates the concept of forward and reverse engineering.

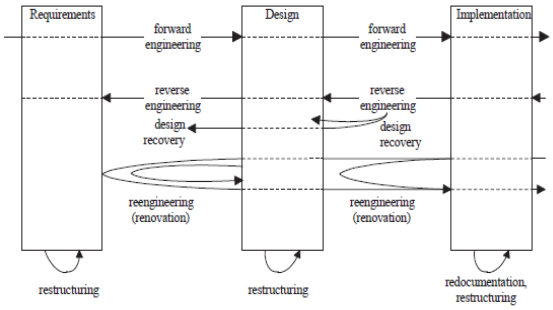

Fig. 1. Forward & Reverse Engineering

A. *Tools and methodologies for software architecture reconstruction:*

Reconstruction of software architecture highlights the significant ways to provide the reconstruction of the architecture of the system and to evaluate the best likely method to reconstruct the system architecture. The techniques explained in this paper are bottom up techniques, top down approaches and hybrid techniques.

B. *Bottom up Techniques:*

In the bottom up technique information gathering for reconstruction the architecture is started from the lower level of gathering facts and aggregate the knowledge to higher levels. Source code analysis is populated in a repository which is inquired to get abstract representations of the system [9] [3]. The process of bottom up techniques is shown in figure 2.

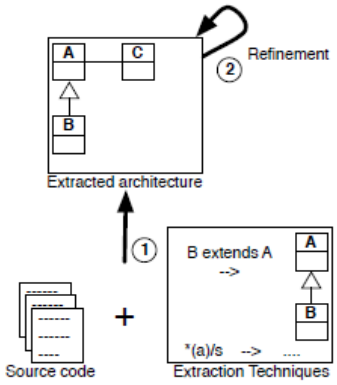

Fig. 2. view extraction from source code (1) and then refinement of extracted views (2)

There are many bottom up techniques but only ARMIN, Dali and Rigi are discussed in this paper due to their good results.

a) Architecture Reconstruction and Mining: ARMIN (Architecture Reconstruction and Mining) is an architecture reconstruction tool developed by the Software Engineering Institute and Robert Bosch Corporation. Once data is gathered, further relationships are then manipulated. This includes collecting, organizing and collapsing. In the end the results can be viewed in an aggregator [10]. The architecture reconstruction method using ARMIN consists of two steps.

- The first step is the source information**:** The elements and their relations are extracted from the system are inserted into ARMIN.
- The second step is architectural view composition: Views of the system's architecture are produced by extracting the source information via aggregation and manipulation. The views are offered to the reconstructor which is present in the ARMIN tool; user can traverse and manipulate them.

The source code and other information are used as input to the tool. The reconstruction process results into the architectural views presented to the user in the view generator component of the tool. The user can manipulate the views according to his requirements and can generate more views. Figure 3 shows the working of ARMIN.

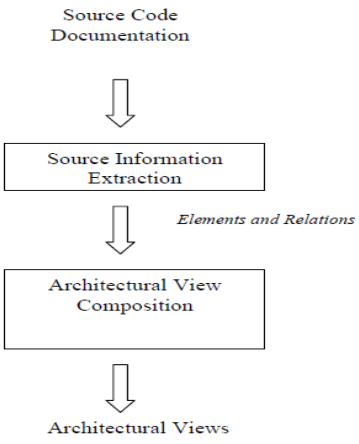

Fig. 3. Architecture Reconstruction Method using ARMIN

The big advantage of ARMIN over the other bottom up software reconstruction techniques is that if more than one views are generated then it will store the previous view also which other tools can't do [11][12][14]. The more detail and usage of ARMIN can be viewed in [13] [14].

b) Rigi: Rigi is a research tool used to understand huge knowledge spaces for example software programs, architecture documentation, and the World Wide Web. This can be achieved by reverse engineering method that models the system by obtaining objects from the knowledge space, managing them into high level abstractions, and representing the graphical model of the given system. [22]. the exact working process of Rigi is shown in figure 4.

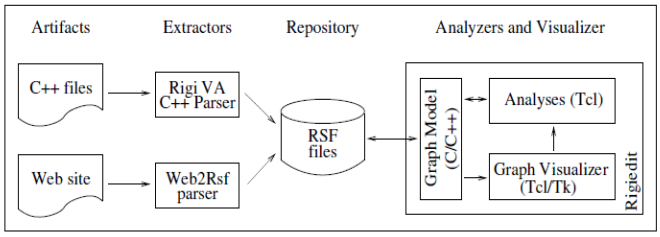

Fig. 4. Rigi Architcture

The main activities of Rigi's architecture include, extraction of facts forms existing systems, a repository which represents and store facts and analyzing and visualizing facts.

- *Fact extraction:* The process of Reverse engineering starts with extracting facts from software's sources. Sources can be inherent artifacts that are essential to compile and build up the system or supporting artifacts. Fact extractor can be constructed for a particular language. This approach can be further divided into two approaches; parser based and lexical extractors. Parsers produce a parse tree without uncertainties. Whereas lexical extractors are constructed on pattern matching of regular expressions [23].
- *Repository data model:* The significant component is the repository. It stores all the facts extracted from the target system. Information stored in the repository is presented to the user with visualizers.
- *Graph-based editor:* The essential part of Rigi is a graph editor, rigiedit. Rigi's functionality is similar to the functionality presented by basic graph editors. Graphs can be loaded, saved, and laid out; the windows depicting a graph can be scrolled and zoomed; the nodes and arcs can be selected, cut, copied, and pasted in a graph; Examples include computation of cyclomatic complexity. Rigi joins graphical visualization with textual reports to offer information about the graphs at different degrees of detail. [23]

c) Dali: The Dali architecture is a structure aimed at to provide combination of an extensive variety of extraction, analysis, manipulation, and presentation tools. In Dali's structural design, rectangles represent different tools and lines depict the data flow among them. The structural design of Dali is shown in figure 5.

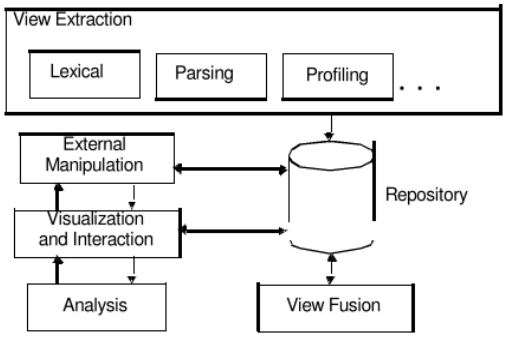

Fig. 5. Dali Structural Design

To extract the source model there are variety of tools like lexical, parser and profiling based tools that generate static and dynamic views of the system under analysis. Static view consists of static source artifacts which are extracted from source code of the system. Dynamic view consists of dynamic elements. These extracted views are then stored in repository which can be relational database. These extracted views are then fused together into *fused views.* In the end, visualization tools are deployed in Dali to present the source model and the result of system architecture analysis. An example of this is Rigi, which can be used to present systems as a graph having nodes which denotes the artifacts and arcs represents the relations between them.[6]

C. Top down approaches:

In these approaches reconstruction is started by previous high level knowledge such as requirements and architectural styles about the application domain and then formulates the hypothesis which is verified against the source code. Figure 6 shows the top down approach of software architecture reconstruction.

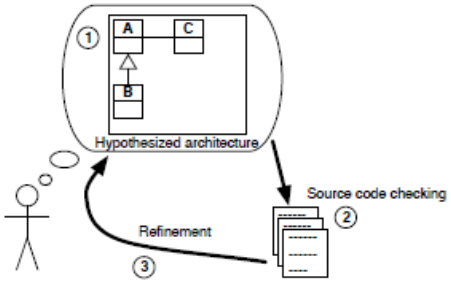

Fig. 6. Top down approach for SAR, hypothesized architecture (1), architecture conformance against source code (2), architecture refinement (3)

The term architecture discovery also defines this process [9] [3]. Following are the top down approaches [9].

- RM Tool
- Pulse
- W4

D. Hybrid approaches:

In this approach, top down and bottom up approaches are taken together for reconstructing the architecture; the low level information is taken as an abstract to refine high level information. They stop architectural erosions [9] [3]. Figure 7 shows the hybrid approach.

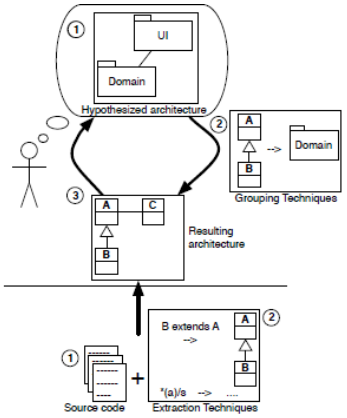

Fig. 7. Hybrid Approach

There are many hybrid approaches hybrid approaches but only cacophony, symphony and Nimeta are discussed in this paper due to their better results [9].

a) Cacophony: It is a Meta model driven architecture reconstruction [8]. The model of a system gives a simplified view of a system. The model should able to answer the queries like the original system. A metamodel is also a model that describes a way of representing the model. Representation of models and Meta models in cacophony is shown in figure 8.

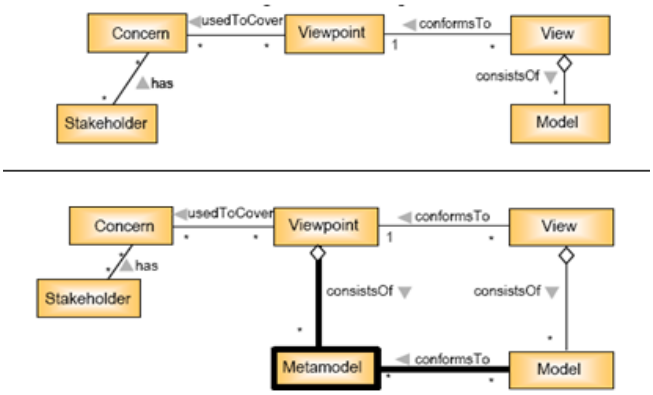

Fig. 8. Models and Metamodels in Cacophony

There are several steps mainly involved in cacophony [17]. In the first step the application domain of the system is analyzed but according to the architectural point of view. In the second step an inventory is maintained in which the information gathered through interviews, slides, various documents etc. is kept a raw mapping between concepts and information is given. In this step the interested information is taken out of the inventory and the conceptual model is developed. The various conceptual Meta models developed from the gathered information stored in the inventory all the metamodels are combined together to make one Meta model. In the next step the metamodel is again analyzed in which it should be kept in mind that the combined metamodels should be clustered in the cohesive way so that when they are needed to be analyzed separately no dependency exists between them. In the next step three things are developed actors identification, use case identification and use case description. Actors are usually the stake holders of the system. Now the stake holders and the use cases are combined so as to view and analyzed where actually the gap is occurs in the system, for this purpose several meetings and interviews are held which specify the problem. Now from the requirements specifications are highlighted and the use case is passed from Meta model. The software is now visualized and in the end its implementation, evaluation and evolution is made.

b) Symphony/Nimeta: In symphony view points and views are used in which are used in constructing the architecture reconstruction models [8].

- *Viewpoints:* These are mostly discussed at abstract level by selecting a set of architectural concepts and rules. It is to be done for focusing on the specific aspect of a system [8].
- *Views:* A view on the basis of given view point gives a representation if a system [8].

E. Views in Symphony:

- *Source View* The view of the system can be getting from source code.

- *Target View* This is the final view that contains the implementation information that is needed to solve the problem.
- *Hypothetical View* It shows the present understanding of the architecture but mostly it is not correct.

There are two stages needed to be fulfilled while reconstruction the architecture [19].

1) In this phase problem elicitation is done by communicating with stake holders and then problem is identified. Then the architectural concepts are revealed related to solve the problem and then a proper recovery strategy for that problem is developed.

2) The specification needs of an architecture reconstruction are viewed in which the source view creates mapping with the target views to solve the identified problem.

III. Software Quality Attributes

Quality attribute requirements specify the nonfunctional requirements of software application, which captures many aspects of how the functional requirements of an application are achieved. Designers need to determine the following points when architecture of the software is specified:

- The amount to which software architecture features can influence the quality attributes.
- The amount to which techniques can support or conflict the attributes.
- The amount to which various qualities attribute requirements can be fulfilled at the same time.

IV. Quality Attributes Driven Software Architecture Reconstruction

In [4], Quality attribute driven evaluation to reconstruct the software architecture is introduced. This technology is used to presents an analysis framework and illustrates the information about the system software. This information is used for the method of reconstruction to relate the knowledge obtained back from organization's business goals. The goal of this approach is to offer extensive information that will contribute to analyze the software quality attributes.

a) Application contexts: Few application contexts in which Software architecture reconstruction can be applied for the analysis of architecture are: [4]

- To streamline current products into product lines.
- To assess the existing systems.
- Decision making between rival existing systems.
- System Reconstruction

b) Quality attributes driven analysis framework: The analysis framework serves as a way to assess systems in the attainment of specific goals of quality attributes, for example scalability and performance goals. The analysis framework serves the architecture reconstruction to make specific characteristics of existing software recognizable. The analysis is driven by business goals, expressed in quality attributes that should be evaluated on existing systems. Evaluations involve a systematic way to reason about the achievement of quality goals. We indicate the systematic way as a framework which helps the software architect to assess or design architectures. The analysis of quality attribute framework is shown in figure 9.

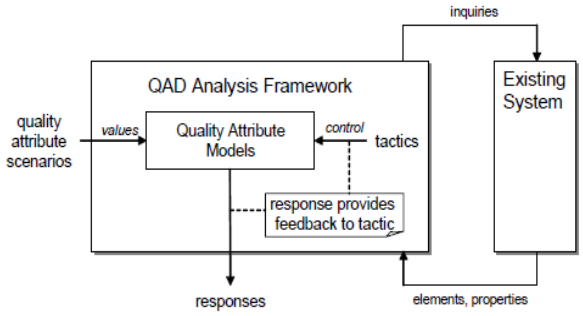

Fig. 9. Analysis framework of Quality Attributes Driven

Quality attributes are improved into quality attribute "scenarios". It is a requirement which is related to quality attribute of the system. It comprises mainly of 1) stimulus and 2) response. The stimulus acts like a signal to the system when the signal reaches the system and then a respective response regarding the stimulus is generated by the system. The quality attribute facts also tell about the generating point of stimulus, which procedure or component generates the stimulus how the response is taken into consideration [15]. A tactic in architectural reconstruction represents the association between design decisions and the response from quality attributes [3]. The Quality Attributes Driven Analysis Framework handles the information extracted from the existing system to be used in Quality Attribute Model with the required architecture elements. Architecture elements, properties, relations, and tactics are integrated under the model of architecture views. An architecture *view represents the set of elements of the system and relationships between them* [16]. The steps of quality attribute driven framework are shown in figure 10.

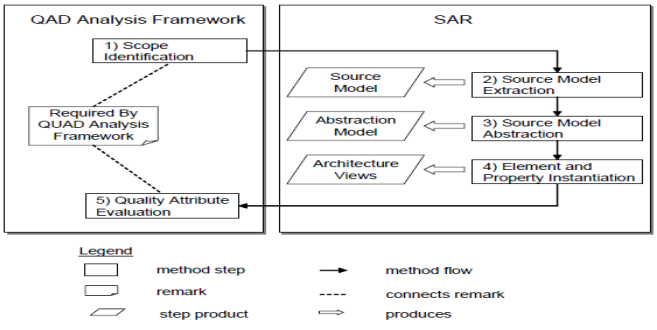

Fig. 10. the QADSAR Steps

There are many phases include in quality attribute software architecture reconstruction. To activate the method, Quality Attributes Driven Analysis Framework needs information about the architecture to perform the quality attribute analysis. Phase 1 defines the scope for Software architecture

reconstruction. The scope recognizes the architecture view types [8] and the system parts that need to be constructed. The identification depends on the quality attribute scenarios, the related quality attribute models, and the type of system.

The phase 2 in the approach involves extracting the elements of source from the resources available. Source elements are the constructs of the implementation language like functions, classes, files, and directories. Relations define how the source elements are related to each other, such as call relations between functions or read accesses by methods on attributes. Besides static characteristics there are also dynamic characteristics like function execution time, or process relations. The static relations are typically generated by existing tools like source code parsers or lexical analyzers. Dynamic information is produced by profiling or code instrumentation techniques. The extracted elements and relations comprise the *Source Model*.

Phase 3 involves identifying and applying aggregation strategies to abstract the detailed views of the sources. There are a lot of strategies for aggregation, which greatly rely on the existing system and the architecture views that need to be extracted. Various techniques exist such as Relation Partition Algebra and Tarski Algebra for manipulation of relational information [13, 14, 19]. The aggregated elements constitute the *Aggregation Model*. The aggregation model consists of entities and relations that are collapsed. They might be associated with architecture elements but they are not explicitly denoted as architecture elements with particular properties.

To acquire the necessary views of the architecture which we assign in phase 4 the types of the elements which are specified by the view-type of the analysis framework. Elements are presented as layers, tasks, 'consist of' relations, etc. We next assign required properties, such as throughput, deadlines for tasks, etc. Further associate tactics are associate that are achieved with a particular set of architecture elements.

The outcomes of step 4 support the QAD Analysis Framework for step 5 *Evaluation Of Quality Attributes* which is performed with the particular quality attribute scenarios, quality attribute models, and the corresponding architecture tactics. The tactics are used to reconstruct the architectural views that helps the quality attribute scenario.

V. INTERFACE IDENTIFICATION:

Interface identification is the reverse engineering technique in which the interfaces involved in the software systems are identified by performing the analysis of source code. It is a bottom up technique in which the components of a system that are externally visible can be identified, some externally visible data elements are also observed. This technique actually shows the interactions of various components. The source code of a system is broken down into small pieces of code and then it is gathered in a way that it should act like a single entity [19].

VI. CLUSTER BASED ARCHITECTURE RECONSTRUCTION:

Clustering approaches are used in many disciplines to provide grouping of related objects of a software system. The basic purpose of clustering exploration is to assist in understanding the observations in a better way and also the construction of complex knowledge structure from features and object clusters. Similar things are grouped into clusters so that similarity between clusters or independency is high, and similarity between different clusters or dependency is low [2]. Clustering algorithms can be divided into two types, namely, partition based and hierarchical.

a) Partition clustering algorithms: Partition algorithms starts with a primary partition consisting of a certain amount of clusters. The partition is then amended at every step and some condition is optimized while keeping the number of clusters constant. Subdivisions of partition algorithms contain graph-theoretic, mode-seeking and mixture resolving algorithms. In Partition algorithms, it is necessary to identify number of clusters in advance, which can create difficulty if we do not have previous information about the data set. Additionally, the partition clustering algorithms are not cost effective because the items are partitioned into clusters and this partition leads to the creation of many clusters which make the algorithm expensive. To overcome the computational complexity of partition algorithms, researchers have proposed heuristic-based approaches to assist software architecture reconstruction.

b) Square error clustering algorithms: Square error algorithm starts with a primary division of the entities in a fixed number of clusters and iteratively shuffles entities between clusters to optimize some clustering measure. This measure denotes the quality of the clustering [21].

c) Graph-theoretic clustering algorithms: Graph-theoretic algorithms are partition algorithms that operate on graphs. The nodes of these graphs correspond to entities and the edges relations between these entities. In general graph algorithms try to split this graph into sub graphs that will form the clusters, instead of focusing on the entities themselves. [20]

d) Hierarchical clustering algorithms: Hierarchical clustering is one of the clustering techniques that are based on a hierarchical breakdown of nodes. Hierarchical algorithms can be further divided into agglomerative and divisive algorithms. In divisive algorithms whole graph is taken as one single cluster initially.in further steps of the algorithms this cluster is divided into smaller clusters in hierarchy until each vertex is denoted by one cluster. Whereas in agglomerative algorithms definition starts with the representation of one cluster for each vertex in the graph. Moving towards next steps in the algorithm, the two clusters having the highest similarity are combined to develop a new cluster [20] [2] when there is only one cluster left the process of agglomerative algorithm stops.

e) Vertex similarity: This function defines the similarities of vertices. There are vertices and edges if two vertices have similar property so they have a strong bonding between them and they will be assigned a higher priority value [21].

VII. COMPARATIVE ANALYSIS

In architecture reconstruction process the top down and bottom up approaches are used these approaches are not only advantageous but also they have many draw backs. The drawback of bottom up approaches is that they are mostly manual, consume much time and they can only work in the particular domain the knowledge used in the specified domains are mostly used as an input knowledge. The drawback of clustering is that these algorithms are automated and their verification is manual. The drawback of top down approach is that they generate many views at a time during exploration and it creates ambiguities in finding the interested views so an interested view can only be found by analyzing each view separately [24].

ARMIN extracted information from the code in rigi standard format [14]. The big advantage of ARMIN over the other bottom up software reconstruction techniques is that if more than one views are generated then it will store the previous view also which other tools can't do [11][12][14].

The information gathered from software is manipulated and visualized using rigi. It contains an interpreter that applies operations on the visuals extracted from the software. The nodes can be selected or removed manually. Parsers are present in rigi that give the extracted information in rigi standard format. Dali is the collection of many tools. Dali is the extension of rigi because in rigi only the visual effect of the extracted information is shown but in Dali queries can be applied on data generated by view of the system. In Dali more than one view are generated at a time. ARMIN is the further extension of Dali. It has the effect of both rigi and Dali but it is advantageous over both the techniques that it not only generates many views bit it can also store the previous views [25].

VIII. CONCLUSION

In this paper we briefly analyze different approaches for software architecture reconstruction process. It is seen that among all approaches bottom up approach is the appropriate approach for reconstruction the architecture because top down and hybrid approaches at certain points leads to the bottom approach. ARMIN is the most appropriate tool for performing architecture reconstruction because it sum up the aspects of all other tools in it and provides an ease of use to the users. After this survey in the future the architecture reconstruction is performed practically using ARMIN.

REFERENCES

[1] Damien Pollet Stéphane Ducasse Loïc Poyet Ilham Alloui Sorana Cîmpan Hervé Verjus Towards A Process-Oriented Software Architecture Reconstruction Taxonomy, LISTIC, University de Savoie, France, july-aug-2009.

[2] Ioana Sora, Gabriel Glodean, Mihai Gligor, Software Architecture Reconstruction: An Approach Based on Combining Graph Clustering and Partitioning. Proceedings of the (ICCC-CONTI), 2010 International Joint Conference on 27-29 May 2010 Department of Computers Politehnica University of Timisoara

[3] Stéphane Ducasse Damien Pollet. Software Architecture Reconstruction: a Process-Oriented Taxonomy July/August 2009 (vol. 35 no. 4)

[4] Christophe Stoermer, liam O'brien , Chris Verhoe, Moving Towards Quality Attribute Driven Software Architecture Reconstruction. Proceeding WCRE '03 Proceedings of the 10th Working Conference on Reverse Engineering

[5] http://www.sei.cmu.edu/architecture/research/previousresearch/reconstruction.cfm. Liam O'Brien Software Engineering Institute Carnegie Mellon University 4500 Fifth Avenue Pittsburgh, Chris Verhoef Free University of Amsterdam De Boelelaan 1081a 1081 HV Amsterdam The Netherlands

[6] Software Architecture Reconstruction René L. Krikhaar.

[7] Claudio Riva, View-based Software Architecture Reconstruction. Institute of Information Systems Distributed Systems Group Vienna, Austria

[8] Vijaya Datta Mayyuri Software Architecture Reconstruction, Symphony, Cacophony. Reverse Engineering, 2004. Proceedings. 11th Working Conference, 8-12 Nov, 2004.

[9] Mircea Lungu ,The 5 questions you always asked yourself about Software Architecture Recovery, Faculty of Informatics, University of Lugano October 2008.

[10] Ian Gorton, Liming Zhu, Tool Support for Just-in-Time Architecture Reconstruction and Evaluation: An Experience Report. Proceedings of the 27th international conference on Software engineering, 2005

[11] Study Liam O'Brien Christoph Stoermer ,Architecture Reconstruction Case, April 2003

[12] Liam O' BrienVorachat Tamarree Architecture Reconstruction of J2EE Applications: Generating Views from the IVIodule View type, November 2003

[13] Rick Kazman Liam O'Brien Chris Verhoef , Architecture Reconstruction Guidelines, Third Edition Publisher, Software Engineering Institute, November 2003.

[14] Liam O'Brien1 Dennis Smith, Grace Lewis Supporting Migration to Services using Software Architecture Reconstruction, Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop

[15] Bass, L.; Clements, P. and Kazman, R. Software Architecture in Practice, third Edition. Publisher, Addison Wesley, October 5, 2012

[16] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R. and Stafford, J., Documenting Software Architectures: Views and Beyond, publisher, Addison Wesley, October 6, 2002.

[17] Jean-Marie Favre, Cacophony Metamodel-Driven Software Architecture Reconstruction, University of Grenoble, France

[18] van Deursen, A. CWI & Delft Univ. of Technol., Netherlands Hofmeister, Christine ; Koschke, R. ; Moonen, L. ; Symphony: View-Driven Software Architecture Reconstruction. Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on12-15 June 2004

[19] Michael W. Barton Richard C. Chapman. A Technique to Identify Component Interfaces.

[20] Chung-Horng Lung , Software Architecture Recovery and Restructuring through Clustering Techniques, ,Software Engineering Analysis Lab, Nortel, Proc. of the 3rd International Software Architecture Workshop (ISAW), 1998

[21] Niels Streekmann, Clustering-Based Support for Software Architecture Restructuring, Publisher, Vieweg+Teubner Verlag; 2011 edition (December 14, 2011)

[22] Kenny Wong , Rigi User's Manual Version 5.4.4 June 30, 1998

[23] Holger M. Kienle Hausi A. Muller, the Rigi Reverse Engineering Environment, University of Victoria, Canada

[24] Mrs. S. Rajeshwari Mr. Telugu Manohar Software Reconstruction Techniques For Software Architecture published in International Journal of Advanced Trends in Computer Science and Engineering, , 2013.

[25] Carnegie Mellon Liam O'Brien Christoph Stoermer Chris Verhoef Software Architecture Reconstruction: Practice Needs and Current Approaches Publisher, Software Engineering Institute August 2002