LITERATURE SURVEY

2.1 Architecture Recovery

The development of program families is important for the recovery of higher level representations from the given source code of an existing software system. The reverse engineering technology \([2,17,26,29,52,54]\) is therefore used to define how architectural elements can be identified in the present software system.

2.2. An understanding to Architecture Recovery

At the present time, software Architecture \([19,66,77]\) is a major area for research. We define Architecture recovery so the new laws will soon replace the existing definition for architecture recovery as a process of identifying and obtaining higher level abstractions from existing software systems. The framework of a system and its architectural representations can be determined by software architectures, which are used by software developers to clearly and fully express, to examine again and manage architectures.

2.3. Architecture Representation

An architectural representation \([12,19,66]\) has as parts structural and non structural information about software architecture. Structural information describes the way that the user prefers the software work of a system and non structural information describes its architectural properties.

Eg: Safety patterns, Communications patterns.

Architecture can be recovered in two ways

1. Manual Approach
2. Using Reverse Engineering Tools

In Manual Approach, the domain experts analyze the source code using Knowledge that is not available for reverse engineering tool. Such knowledge includes information about domain knowledge, high-level design decisions, coding standards and system requirements. The manual views are recorded so that these views are used for the first level of abstractions after analyzing the source code in the architecture recovery process. But yet, these views do not contain architectural information. The reverse engineering community [13,17] uses the software matching approaches in composing the pattern to provide a cooperative instrument for architectural recovery to use domain knowledge and to view architectural information. In this way Reverse engineering tools [19,54] are used together with application domain knowledge to complete the recovered software views by associating them with manually generated tools.

2.4 Pattern based framework

A query is represented as a high level, view of a software system in terms of system components and their instructions. It is mapped on to a pattern graph, where a module and its interaction with other modules are represented as a group of graph nodes and group of graph edges respectively such as pattern graph is applied against an entity relation graph that represents the information extracted from the source code of the software system.

2.5 Architecture view

Architecture views serve as examples of the overall architecture that are meaningful to one or more stakeholders in the system. The architect chooses and develops a set of views that will enable the architecture to be communicated to
and understood by, all the stakeholders, and make them able to verify that the system will addresses their concerns.

A view comprises selected parts of one or more models, so as to demonstrate to a particular stakeholders or group of stakeholders that their concerns are being adequately addressed in the design of the system architecture.

The architecture views are of the following categories

2.5.1 Business Architecture views

It directs the concerns of users, planners and business managers, focus on the functional aspects of the system from the perspective of the users of the system, which is on what the new system is meant to do, including performance, functionality, and usability.

These can be built up from an analysis of the existing requirement and of requirement and constraints affecting the new system.

- The people views focus on the human resource aspects of the system. It examines the human actors involved in the system.
- The Business process view deals with the user processes involved in the system
- The Business function view deals with the functions required to support the process.
- The Business Information view deals with the information required to move along in support of the processes.
- The usability view takes into account the usability aspects of the system and its environment
- The Business performance view takes into account the performance aspects of the system and its environment.
2.5.2 Data Architecture views

It addresses the concerns of database designers and database administrators, and system engineers for developing and combining into a whole the various database components of the system.

2.5.3 Application Architecture Views

It directs a message to the concerns of system and software engineers of system and software engineers responsible for developing and interrogating the various application software components of the system.

Database designers, database administrators, system engineers and software engineers focuses on how the system can be carried into effect from the perspective of different types of engineers and how that affects its properties. System and software engineers are typically concerned with modifiability re-usability and availability of other servers.

2.5.4 The Data Flow view

It deals with the architecture kept ready for the future use, retrieval, processing, archiving and safety of data. It looks at the flow of data as it is stored and processed, and what components will be required to support and manage both storage and processing.

2.5.5 The software Engineering View

It deals with aspects of interest to software developers. It takes into account what software development acts under opportunities exist in the new system, and looks at how development can be carried out, both in terms of technology and resources. The software engineering view is particularly important, because it provides a reference for selection of building blocks related to elements of the existing system that may be used again in the target Architecture.
2.5.6. The System Engineering View

It offers a number of different ways in which software and hardware components can be assembled into a working system. The choice of model determines the properties of the final system to a great extent. It works at technology which is already present in the organization and what is available currently or in the near future. This makes known areas where new technology can contribute to the function of efficiency of the new architecture, and how different types of processing platform can support different parts of the overall system.

2.5.7 Technology Architecture View

It addresses the concerns of acquirers, system administrators, operators, communication engineers and managers of the system.

a. The Communications Engineering View

It addresses the concerns of the communications engineers. It puts questions to in order to test various ways of structuring communications facilities to understand easily, the business of network planning and design.

It examines the networking elements of the architecture in the light of geographic constraints, bandwidth requirements and so on.

b. Acquires Views

It addresses the needs of an acquirer or procurer, providing suitable guidance for producing components that fit the architecture.

Acquires views of the architecture are primarily concerned with costs and standards that must be adhered to

For example:

(i) The cost view.

(ii) The standards view.
These views being the representations depict building blocks of the architecture that can be purchased and the standards that the building blocks must adhere to in order for the building block to be most useful.

2.5.8 Composite Views

- The Enterprise Manageability view addresses the concerns of the operations, administration, and management of the system, and Concentrates more on the details of location, type, and power of the equipment and software in order to manage the health and availability of the system. It covers issues such as initial deployment, upgrading, availability, security, performance, asset management, fault and event management of system components, from the management perspective of the following subject matters:
  - Security
  - Software
  - Data
  - Computing/Hardware
  - Communications

- The Enterprise Security view focuses on the security aspects of the system for the protection of information within the organization. It examines the system to establish what information is stored and processed, how valuable it is, what threats exist, and how they can be addressed.

Architects must understand the relationships between those views, and dealing with the conflicts that arise from those different views. Architects also must deal with viability of the architecture: if the architecture is not capable of being implemented, then its value is in doubt.
Examples of specific views that may be created in each category are tabulated below.

<table>
<thead>
<tr>
<th>To address the concerns of the following stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users, Planners, Business Management</td>
</tr>
<tr>
<td>Database Designers and Administrators, System Engineers</td>
</tr>
<tr>
<td>System and Software Engineers</td>
</tr>
<tr>
<td>Acquirers, Operators, Administrators, Managers</td>
</tr>
</tbody>
</table>

.....the following views may be developed

- **Business Architecture Views**
  - Data Architecture Views
  - Applications Architecture Views
  - Technology Architecture Views
- **Business Function view**
- **Business Services view**
- **Business Process view**
- **Business Information view**
  - Data Entry View
  - Software Engineering View
- **Business Locations view**
- **Business Logistics view**
- **People View (Organization Chart)**
  - Data Flow View (Organization data Use)
- **Working flow View**
- **Usability View**
- **Business Strategy and Goals View**
  - Logical data View
- **Business Objectives View**
- **Business Rules View**
  - Software Distribution View
- **Business Events View**
- **Business Performance View**
  - Cost View
  - Standards View
- **System Engineering View**
- **Enterprise Security View**
- **Enterprise Manageability View**
- **Enterprise Quality of Service View**
- **Enterprise Mobility View**

In order to achieve the goals of completeness and integrity in architecture, architecture views are usually developed, visualized, communicated, and managed.
2.6 Concept Lattice Analysis

In this context, the lattice is referred to as a binary relation between a set of objects and a set of attributes. Reverse Engineering uses the concept of Lattice analysis [3,15,50] now days. This uses a maximum collection of System functions (objects) which shares maximum common attributes. The major importance is to provide relationship between objects and their attribute values such that each node represents a lattice.

Sometimes the illustration of this lattice is not known. Then concept lattice becomes complex in software systems. In that case some algorithms are used for partitioning to aid the user to find distinct clusters.

2.6.1 Steps for partitioning the system

1. A matrix of functions and their attribute values are set.

2. A concept lattice is constructed based on the matrix that was built using bottom-up iterative process.

3. A collection of the concept partitions is recognized. Each partition should be a group of disjoint sets of concepts and there should be a large overlap in each attribute values set.

4. Finally each partition corresponds to a probable corrosion/decomposition of the system into modules finally

There are 2 techniques proposed by Sif [50] and Snelting [15] for making the partitions easier.
1. Repair technique

2. Horizontal decomposition technique

**Repair Technique:** To provide easier partitioning, it adds extra relations to the generated concept lattice. Due to this very huge number of partitions will be generated. So user interface is more. We consider this as a drawback for this proposed technique.

**Horizontal decomposition technique:** In this technique a lattice of procedures and variables are partitioned into modules. Such a horizontal partitioning can be prevented by the number of interferences between concepts in the lattice of a real system.

2.7 Clustering

Clustering is the assignment of objects into groups (called clusters) so that objects from the same cluster are more similar to each other than objects from different clusters. The key concept of clustering is to group similar things into clusters, such that intra-cluster similarity or cohesion is high, and inter-cluster similar or coupling is low.

Clustering is useful in several exploratory pattern-analysis, grouping, decision-making and machine learning situations including document retrieval, image segmentation and pattern classification. In most of the cases there is little aforementioned information available about the data and the decision maker must make as few assumptions about the data as possible. So, Clustering is particularly useful for exploration of interrelationships among the data.
Clustering techniques [3,36,58] have been used in many disciplines to support grouping of similar objects of a system. Clustering analysis is one of the most fundamental techniques adopted in science and engineering. The main objective of Clustering is similar to that of software partitioning. Clustering techniques can be applied to software during various life-cycle phases Clustering techniques can be used to effectively support both software architecture partitioning [32,59,76] at the early phase in the forward engineering process and software architecture recovery of legacy systems in the reverse engineering process. Many Clustering algorithms comprise the following three common key steps:

- Obtain the data set.
- Compute the resemblance coefficients for the data set.
- Execute the Clustering method.

An input data set is an object-attribute data matrix. Objects are the entities that we want to group based on their similarities. Attributes are the properties of the objects. A resemblance coefficient for a given pair of objects shows the degree of similarity or dissimilarity between these two objects, depending on the way the data represents. Given a resemblance matrix, a Clustering method is then used to group similar objects. A commonly used Clustering algorithm called UPGMA (UN weighted pair-group method using arithmetic averages) is used to cluster the objects. To apply the Clustering techniques to software architecture recovery and reengineering, the object-attribute data matrix is modified to object-object data matrix, so that the input reflects the interconnectivity of components. The Clustering techniques are then used to minimize interconnections among components.
2.8 Graph Matching

I am presenting a graph-matching model for the software architecture recovery problem. Because of their expressiveness, the graphs have been widely used for representing both the software system and its high-level view.

Modeling the recovery process as graph matching is an attempt to identify a sub-optimal transformation from a pattern graph, representing the high-level view of the system, onto a sub graph of the software system graph. A successful match yields a restructured system that conforms to the given pattern graph. A failed match indicates the points where the system violates specific constraints. The pattern graph generation and the incrementally of the recovery process are the important issues to be addressed. Initially the graph matching process consists of two major phases:

(i) Restricting the search space of the graph matching by pre-processing the entire input graph and producing a database of graph regions, each loosely satisfying a desired property; and

(ii) Applying a graph-matching algorithm [11,32,40] that approximately matches an architectural pattern, represented as a query, against the database of graph regions.

The matching process computes a sub-optimal match between a pattern graph that originates from an AQL query and an input-graph that originates from the system source-graph. The matching is performed in n phases.

Initially, the first nodes are of the pattern region are selected as the source nodes, however during the matching process the common source node of an edge-bundle that is not matched yet can be redirected to another node without
any cost. We use the search algorithm to generate a search-tree that corresponds to the recovery of each module by using AQL query specification that consists of
i) A root node for matching the main-seed of the source-region
ii) A number of non-leaf tree-nodes at different levels of the search tree that correspond to different alternative matching of the placeholders in the pattern region with nodes in the source region and
iii) Leaf tree-nodes that correspond to solution paths where the placeholders have been instantiated and constrains have been met.

2.9 Fiedler vector
The graph-partitioning problem in mathematics consists of dividing a graph [62,81] into pieces, such that the pieces are of about the same size and there are few connections between the pieces.

Consider a graph \( G(V, E) \), where \( V \) denotes the set of vertices and \( E \) the set of edges. The standard (unweighted) version of the graph partition problem is: Given \( G \) and an integer \( k > 1 \), partition \( V \) into \( k \) parts (subsets) \( V_1, V_2, \ldots, V_k \) such that the parts are disjoint and have equal size, and the number of edges with endpoints in different parts is minimized. In practical applications, a small imbalance \( \epsilon \) in the part sizes is usually allowed, and the balance criterion is

\[
\max_i |V_i| \leq (1 + \epsilon)|V|/k.
\]

In the general (weighted) version, both vertices and edges can be weighted. The graph-partitioning problem then consists of dividing \( G \) into \( k \) disjoint parts such that the parts have approximately equal weight and the size of the edge cuts is minimized. The size of a cut is the sum of the weights of the edges contained in
it, while the weight of a part is the sum of the weights of the vertices in the part.

When \( k=2 \) the problem is also referred to as the Graph Bisection Problem.

A common extension is to hyper graphs where an edge can connect more than two vertices. A hyper edge is not cut if all vertices are in one partition, and cut exactly once otherwise, no matter how many vertices are on each side. This usage is common in Electronic design automation. Graph partitioning [43,62] is known to be NP-complete, but can be solved in polynomial time for \(|V_i|=2\) by matching (see Michael R. Garey and David S. Johnson. Computers and Intractability. Fast heuristics work well in practice. An important application of graph partitioning is load balancing for parallel computing, where each vertex corresponds to a computational task and the edges correspond to data dependencies. A more general model, hyper graphs partitioning, is now often preferred. Software for graph partitioning is widely available.

The original theory related to algebraic connectivity was produced by Miroslav Fiedler [43,81]. In his honor the eigen vector [5] is associated with the algebraic connectivity has been named the Fiedler vector. The algebraic connectivity of a graph \( G \) is the second-smallest eigenvalue of the Laplacian matrix of \( G \). This eigenvalue [5] is greater than 0 if and only if \( G \) is a connected graph. This is a corollary to the fact that the number of times appears as an eigenvalue in the Laplacian is the number of connected components in the graph. The magnitude of this value reflects how well connected the overall graph is, and has been used in analyzing the synchronizability of networks.