A FRAMEWORK FOR SOFTWARE ARCHITECTURE VISUALIZATION AND EVALUATION

Dr. S. Margret Anouncia
Professor,
School of Computing Sciences
VITU, Vellore

Merin Cherian
M.S Software Engg
VITU, Vellore

Anubhuti Parija
M.S Software Engg
VITU, Vellore

Dulcy Sylvia.R
M.S Software Engg.
VITU, Vellore

Jayaprasanna. D
M.S Software Engg
VITU, Vellore

Abstract

This paper presents a framework for visualization and evaluation of software architectural styles. There has been significant research made to improve the software architecture visualization and evaluation. Most of the tools developed for this purpose don't satisfy all the framework's elements. Hence the paper presents a framework that builds modules from requirements, measure modularity, visualizes architecture and evaluates the visualized architecture satisfying all elements.

INTRODUCTION

Software design is the process of applying various techniques and principles for the purpose of defining a system in sufficient detail to permit its physical realization. Visualization makes people to understand information presented in a shorter time or in a great depth. The output of the design process is the design description. As the design description is complex and difficult to understand, there is a need for visualization method for better understanding. There is no existing tool for proper visualization.

1. MOTIVATION OF THE PAPER

Currently the most challenging problem is the transition from software requirement to appropriate architecture design of software system. Most of the requirements are conflicting and unpredictable in nature. The absence of a proper automated tool which can evaluate all the attributes for an architectural style also adds to need for further research in design field. Manually designed architectural styles are misleading and time consuming. Hence there is a necessity for an automated tool which should generate the appropriate architecture and its evaluation.

2. RELATED WORK

The automated transformation of software requirements into architectural design is one of the challenging fields. A lot of research is being performed throughout. Some important ones are specified below.

The main contribution of [1] by Koen Yskout, Riccardo Scandariato, Bart De Win, Wouter Joosen DistriNet, Katholieke Universiteit Leuven, Belgium is the elaboration of a set of transformations for some important security requirements, namely delegation, authorization, and auditing. These transformations are based on an extensible meta-model capturing the requirements-level concepts that are important for transformation purposes. The second approach which falls under this category is developed by Jorge Enrique Perez Martinez and Almudene Sierra Alonso proposed an automated methodology for the transition from analysis to architecture styles using UML notations. [2].Another research group by

Hassan Reza, Dan Jurgens, Jamie White, Jason Anderson, and Jay Peterson developed a tool based on a set of scenarios that allows the user to select an architecture based on non-functional requirements [3]. Non-functional requirements are then mapped to tactics using weighting (or scoring techniques). The architecture is then selected by its compatibility. Researchers G.Zayaraz and P.Thambidurai proposed a framework for choosing appropriate software architecture based on the quality requirements of different stakeholders [4].

A software architecture design provides a high-level abstraction of system topology, functionality and behaviour. It is source for early system understanding and analysis. It also provides the foundation for subsequent detailed design and implementation. .Researchers Keith Gallagher, Andrew Hatch and Malcolm Munro proposed an approach focusing on the improvement of software architecture visualization using qualitative framework. The main objective is to compare and evaluate the different software architecture visualization tools using the key features of framework. The framework is derived by the application of the Goal Question Metric paradigm called GQM framework [2]..Another approach by Liming Zhu, Muhammad Ali Babar and Ross Jeffery improves the software architecture evaluation process by systematic extraction and appropriate documentation of architecture significant information[6]. Researchers Muhammad Ali Babar, Liming Zhu and Ross Jeffery describes a set of features for evaluation method which provides guidance for selecting the most appropriate evaluation method.[5].

3. OUR APPROACH

Software Architecture defines the overview of the system which consists of various components and their relationship among them. There has been a lot of demand for quality software system which can be primarily achieved through architectural design. Hence this paper proposes a framework for a tool which is named as ‘Architecture Visualization and Evaluation for Software Systems’ (AVESS) .

3.1. OVERALL FRAMEWORK

The architectural design adopted for the proposed framework is pipe and filter which defines a continuous flow of information. Pipe and filter architectural style comprises of components and connectors. Each component has a set of inputs and a set of outputs. A component reads stream of data on its input and produces a stream of data on its output by applying transformation to the input. Components are called filters and connectors are called pipes.

The figure 3.1 explains the architectural design of the proposed framework. The architecture design comprises of eight modules. Each module performs distinct functions required for visualization and evaluation of architectural styles. The requirement extractor module extracts the requirements from the functional requirements given by the user by comparing with predefined requirement keywords. The successor module, module builder groups the requirements based on some predefined criteria and builds modules. The modules are assigned names by user. The generated modules are further refined by measuring modularity which consists of cohesion and coupling. The architecture of the application is determined using some predefined questions. Then the appropriate architecture diagram is generated by the tool. The needed attributes are determined from the extracted requirement features. These needed attributes are compared with predefined attributes of architecture. Each attribute is assigned a value. Then total weight is calculated by summing up the individual attribute weights. Finally the evaluation result is displayed as bar chart.

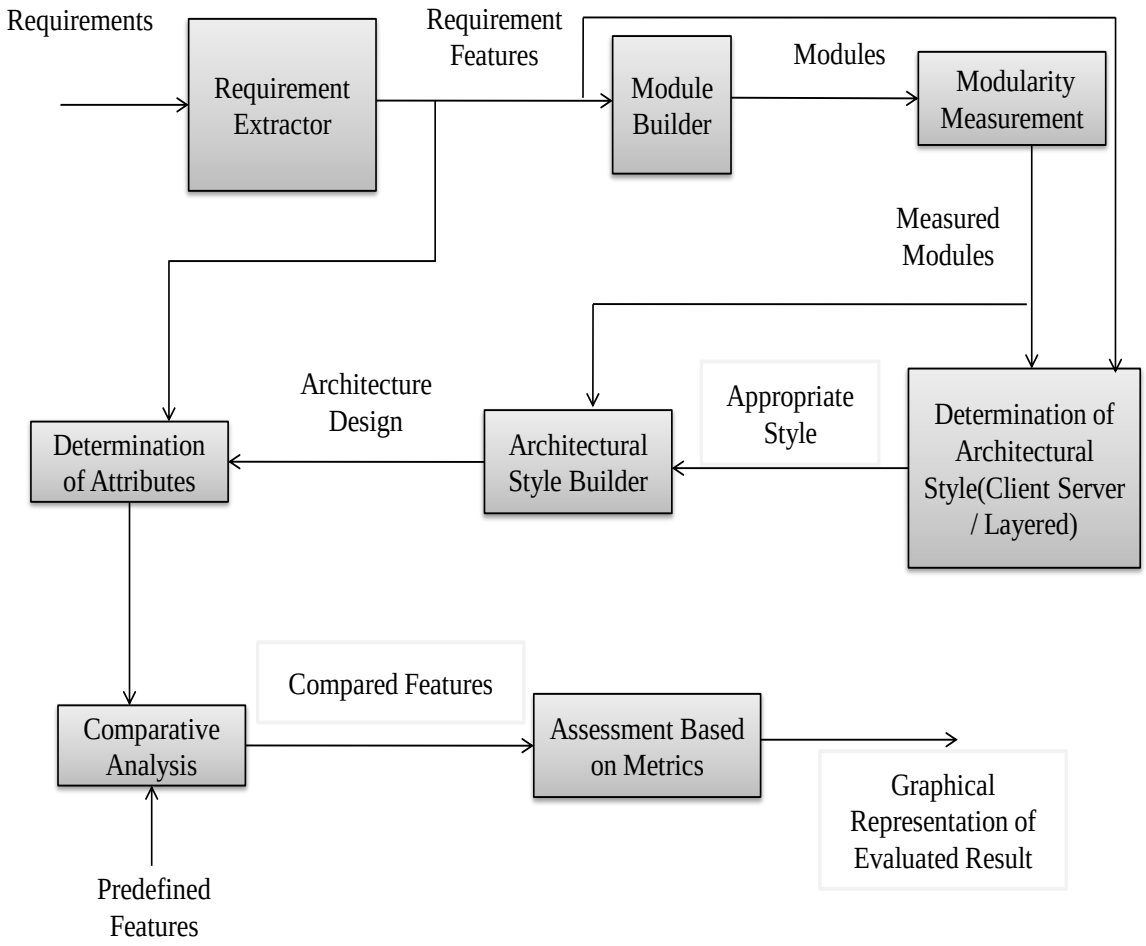


Fig.3.1 Overall Framework

3.2 Use Case Model for AVESS

The overall functionality of this automated software architecture visualization system is depicted using Use Case Model as shown in fig.3.2. This describes a high level process of what an actor will do with a system. An actor may perform an event to start the system. This description does not represent individual steps in the process but represents the high level process itself.

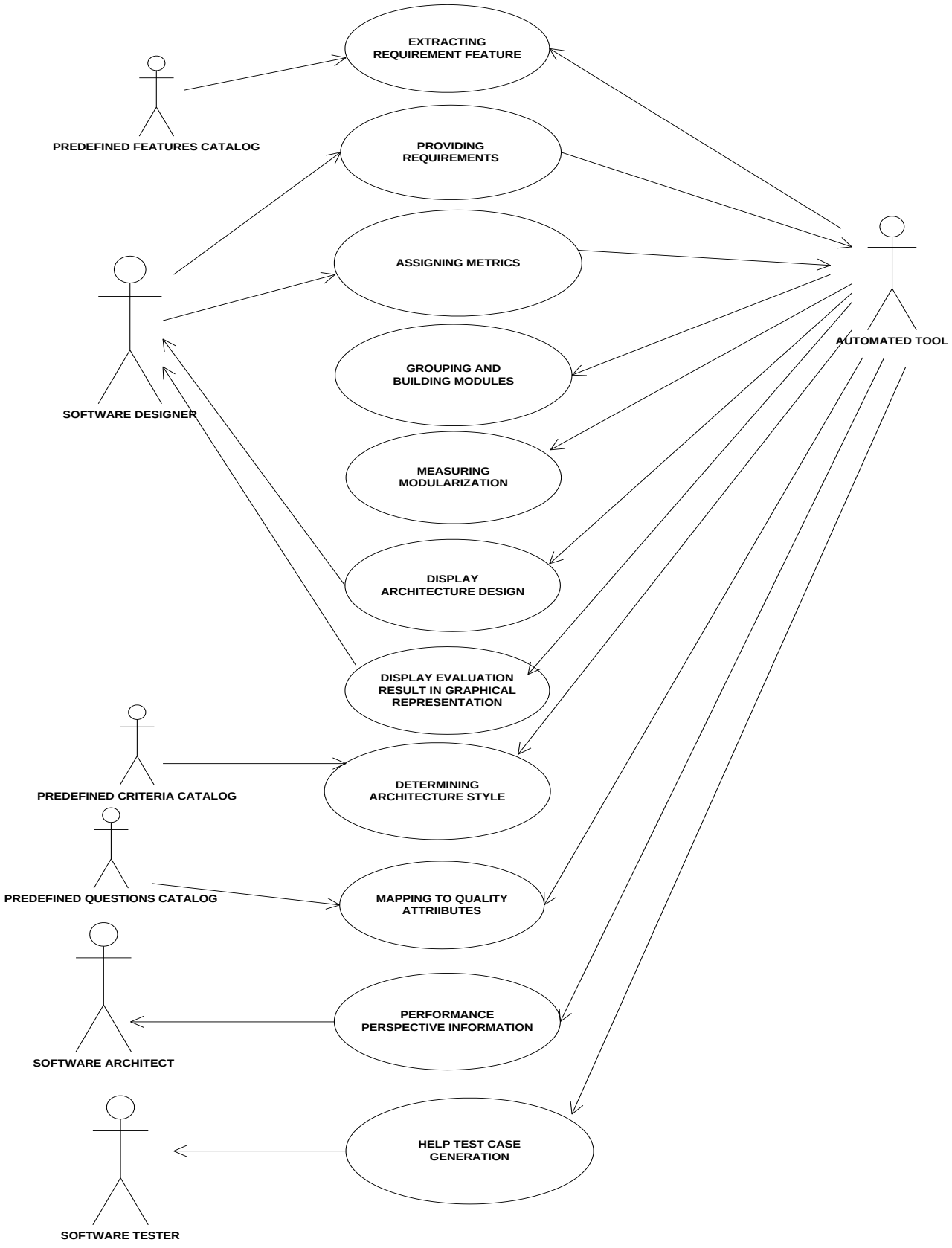


Fig.3.2 Use Case Model for AVESS

3.3. DETAILED DESIGN

On analyzing the given requirements, it was found that data flow diagram (DFD) is the most appropriate model to be used. The level 4 gives detailed design. Context level DFD (Fig.2), Level 1(Fig.3), level 2(Fig.4), level 3(Fig.5) shows

the systems decompositions and finally level 4(Fig.6) brings out the detailed design in which each processing bubble performs a unit function. Context level depicts the input and the output of the system. When the level oriented decomposition is made, the bubbles in the DFD are disintegrated to unit level functions. The complete detailed design is clearly depicted in the level 4 DFD.

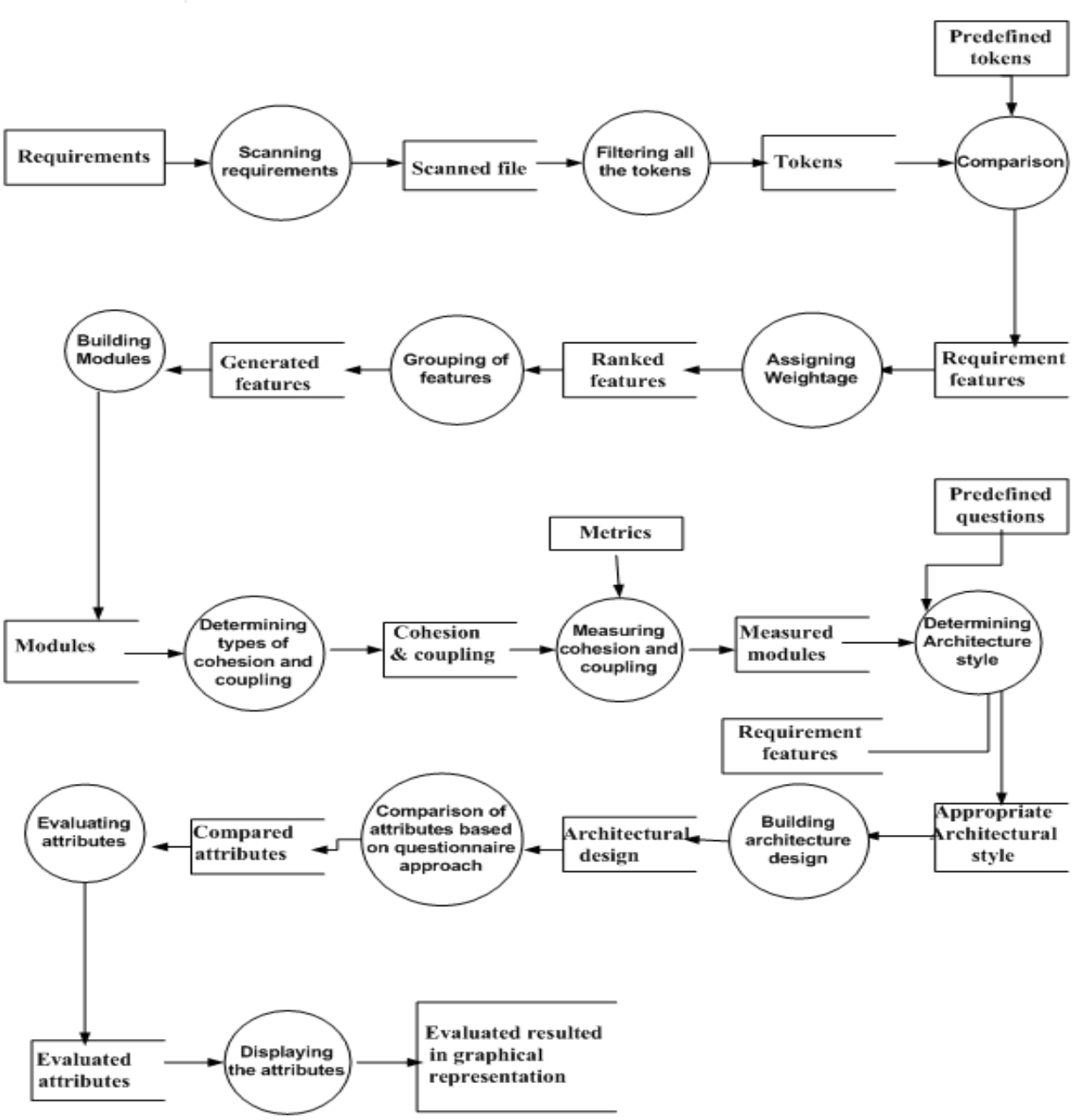


Figure 3.3:4th level DFD

The functional and non-functional requirements are given as input by the user to the automated tool. These requirements are stored in the database. The required features are retrieved from database .These

required features are grouped based on the comparison with predefined keywords. These grouped features are ranked and modules are built and naming of these built modules is done by the user. The interaction among the modules is determined and the modules are measured based on cohesion and coupling. The most appropriate architectural style is determined based on the measured modules and grouped features. The user visualises the appropriate architectural style. The quality attributes are determined from the non-functional requirements. There is a comparison of these attributes with predefined attributes based on questionnaire approach. The compared attributes are evaluated based on metrics. The evaluated attributes are displayed in form of graphical representation.

3.4 IMPLEMENTATION OF FRAMEWORK

A novel visualization and evaluation technique framework is developed for most widely used architecture style. The chapter discusses on the tool which automatically generates and evaluates the architecture from requirements. The implementation of each unit function within the system is described in detail.

3.4.1 TOOLS USED

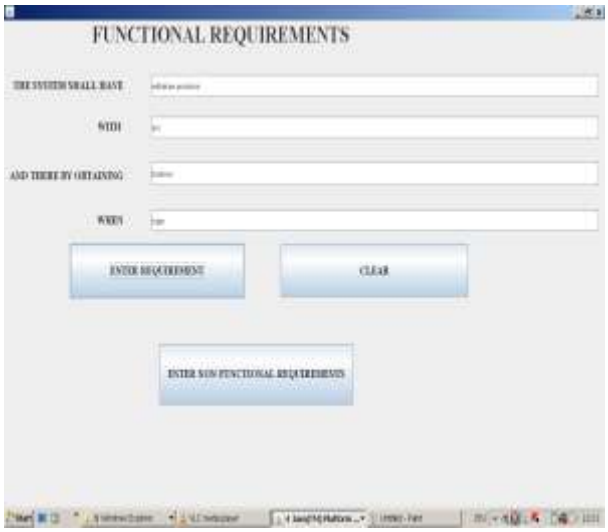
The entire software architecture visualization and evaluation system is implemented using java with NetBeans 6.1. The system is decided to be built using java because it is platform independent as well as it holds many inbuilt functions and string manipulation operations. The visualization of the architecture style can be easily made using java frames.

3.4.2 METHODOLOGY

The implementation of three major modules of the system is described in detail

3.4.2.1 REQUIREMENT EXTRACTOR

The requirement extractor module gets functional requirements as input from the users, extracts the required features based on comparison with predefined keywords and stored in a database. The functional requirements, input, output and dependency considering the requirements are provided by the user in the text box. These requirements are then compared with the predefined keywords stored in an array. Along with this a table called freq is created in the database. Further extracted features along with input, output and dependency are stored in freq table.



3.4.2.2 MODULE BUILDER

The module builder groups the required features based on the dependency of the extracted features and those grouped features are considered as modules. It also allows users to rename the modules. The features along with dependency are fetched fro freq table. These features are then compared with set of keywords stored in different arrays. Some of key arrays

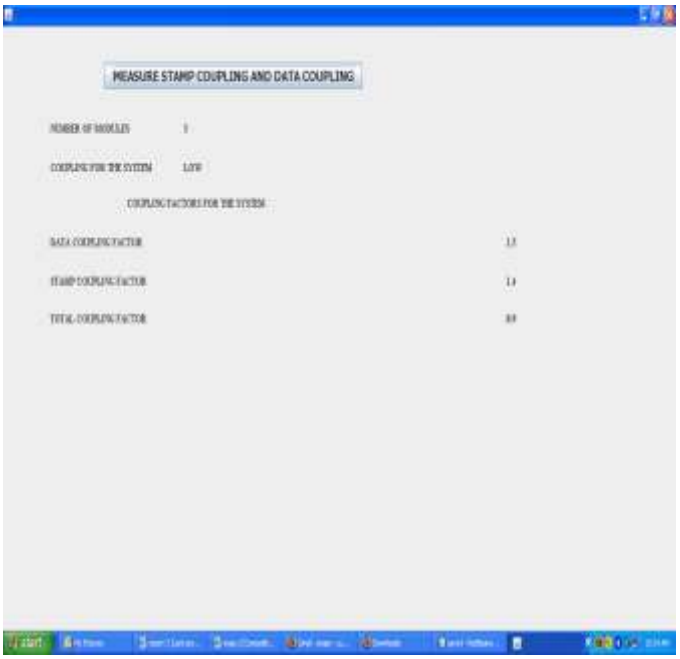
are processing, display, authentication etc. If the dependency is null, then the requirement is considered as separate module. A module will have

many requirements and is considered as different module, only if those requirements have the same

dependency and belong to the same array. Otherwise they are considered as distinct modules. These modules are stored in modfun table along with output, input, link and module function. Then these modules are displayed in rectangular boxes along with module functionality. The user can rename these modules. These renamed modules are stored in module table in database. The module count is also found and stored
for further usage.

3.4.2.3 MODULARITY MEASUREMENT

This module refines modules by measuring cohesion and coupling. The modules along with its fan-in / fan-out and module count are retrieved from database. The modularity is calculated with the help of fan-in and fan-out of each module. i.e. by finding out the dependency between the fan-in and fan-out of the module The type of fan-in and fan-out is determined by finding whether it contains data or data structure. Data coupling is found out by finding whether the data is passed between the modules and Stamp coupling is found out by finding whether the data structure is passed between the modules. Then the overall coupling factor is calculated by applying metrics for the data coupling and stamp coupling.



4. CONCLUSION

Software Architecture defines the overview of the system. This paper gives the proposal of the framework of a tool which visualizes as well as evaluates Software Architectural Styles. The tool will generate architectural design containing Client Server Style and Layered style. So which ever application given, it will either generate client server or layered style. To evaluate the architectural style there is a consideration of quality attributes and framework elements. In future there can be a lot of improvement in the evaluation criteria.

REFERENCES

1**1) Koen Yskout, Riccardo Scandariato, Bart De Win, Wouter Joosen,** *"Transforming Security Requirements into Architecture," ares*, pp. 1421-1428, 2008 Third International Conference on Availability, Reliability and Security, 2008.

1**2) Keith Gallagher, Andrew Hatch, Malcolm Munro,** "*Software Architecture Visualization: An Evaluation Framework and Its Application,"* IEEE Transactions on Software Engineering, vol. 34, no. 2, pp. 260-270, Mar/Apr, 2008
2

1**3) Hassan Reza,Dan Jurgens,Janie White,Jason Anderson,and Jay Peterson:,** *"An Architectural design Selection Tool Design Based on design Tactics, Scenarios and Non Functional Requirements* ",eit, p. 6, 2005 IEEE International Conference on Electro Information Technology (EIT'05), 2005

1**4) G.Zayaraz and P.Thambidurai** "*Software Architecture Selection Framework Based on Quality Attributes",* pp. 167-170, IEEE Indicon Conference, 2005.

1**5) Muhammad Ali Babar, Liming Zhu, Ross Jeffery,** "*A Framework for Classifying and Comparing Software Architecture Evaluation Methods," aswec*, p. 309, 2004 Australian Software Engineering Conference (ASWEC'04), 2004

1**6) Liming Zhu, Muhammad Ali Babar, Ross Jeffery,** *"Mining Patterns to Support Software Architecture Evaluation*," *wicsa*, p. 25, Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04), 2004

1**7) Jorge Enrique Perez Martinez and Almudene Sierra Alonso** *"Heuristics for the transition from Analysis to Software Architecture"* , p. 311**,**Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture **,**2004