A Survey on the Foundation of Software Architecture

Tanzeela Khanam
MBA-IT Student
SICSR, affiliated to Symbiosis International University(SIU), Pune, Maharashtra, India

Tejasvi Chalasani
MBA-IT Student
SICSR, affiliated to Symbiosis International University(SIU), Pune, Maharashtra, India

Tejas Mhaske
MBA-IT Student
SICSR, affiliated to Symbiosis International University(SIU), Pune, Maharashtra, India

Hema Gaikwad
Associate Professor
SICSR, affiliated to Symbiosis International University(SIU), Pune, Maharashtra, India

Abstract

Software architecture research has emerged over the past decade, as the fundamental study of the overall structure of software systems, particularly the relations among subsystems and components. Building the foundation for software architecture is the main focus of this paper. The paper started with developing an intuition for software architecture by appealing to several well-established architectural disciplines. Considering this intuition, a model of software architecture is presented that comprises of three components: elements, form, and rationale. The paper provides a classification of software architectures which turn out to be the foundation for the establishment of marketplaces for software components. The basis of component marketplace lies in the framework of key properties of software architecture. We can understand the development and scenario of software architecture research by examining the research paradigms used to establish its results.

Keywords- Software Architecture, Subsystem, hardware architecture, network architecture, Architectural Styles, Software elements

I. INTRODUCTION

Software architecture is the framework to define the technical and operational requirements of the user. It is a high level structure of any software system, which comprises of software system elements: input, processes, environment, output and boundaries or interface. Mary Shaw explained the overall structure of software systems, especially the relations among subsystems and components. He provided a categorization of software architectures which turn out to be the basis for the establishment of marketplaces for software elements. He worked on the three phases of maturation of software architecture. i) Concept formulation ii)internal enhancement and exploration. iii) External enhancement and exploration. The process of defining structures solution that meets all of the operational and technical requirements, along with the optimization of quality attributes such as performance, security, manageability. The foundation for the study of large scale structures of software systems arose in year 1980.The software engineering research was purely focused on transforming from software design to integrating the designs and further into the broader context of the software design process and its management.

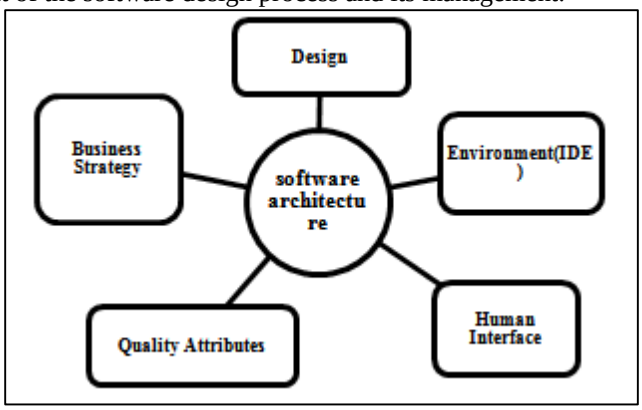

Fig. 1: Software Architecture Elements

In1990's, the term software architecture was referred in contrast to software design .While there has been some work in defining particular software architecture and general support for the process of building architectures.

Some of the benefits that came from the emergence of software architecture as a major discipline are:

1) Architecture as the framework for satisfying requirements;
2) Architecture as the technical source for design and as the managerial source for cost estimation and process management;
3) Architecture as an effective basis for reuse; and
4) Architecture as the basis for dependency and consistency analysis.

The primary objective of our research is to support the use of software architecture specifications in development. We started with Section II by analyzing the problems and challenges in software architecture design. In section III we develop an intuition about software architecture against the setting of well-established disciplines such as network, and building architecture, hardware. Then in section IV we established the context propose a model for, characterization of software architecture and software architectural styles of software architecture. Further in section VI, we elaborated architectural styles.

II. PROBLEMS AND CHALLENGES

Software design begins with one or several box-and-lines drawings said to describe the system's architecture, as shown in Figure 1. But such descriptions are too informal to be of any use for others than the author [11]. What do these different arrows shapes and lines represent? Are they representing data flow or control flow? Do the ellipses and rectangles represent classes, objects, processes, or functions? If the person has the same background and understanding, stakeholder will probably understand the description in much the same and easier way as the author of it does. But if there is there is greater difference between their experiences, then there will be the higher probability that the reader will misinterpret the concept and architectural description, or be unable to interpret it at all. If there is more formal description then there would be less possible sources of misinterpretations; moreover it would be possible to carry out many analyses automatically, such as validation and simulation. There are also more informal methods to the art of engineering software architecture, which already has proven successful.

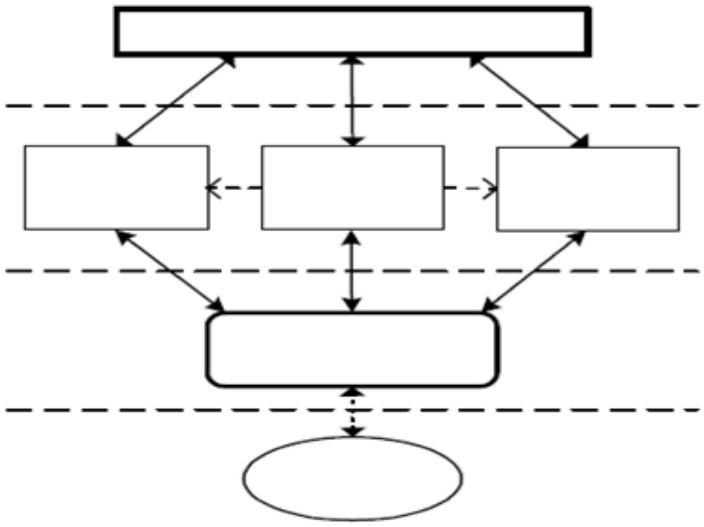

Fig. 2: Informal Description of Software Architecture

The following changes are still noticeable in software architectural design:

A. Shape Descriptions (Box and Line)

There are two reasons why the visualization picture doesn't work well as an architectural design: firstly, it shows too much of information in a single representation, and secondly, no one is really sure about the different types of symbol that an architect have drawn mean."

B. Processes

A broad set of activities and tasks needs to be followed in the design phase that links the gap between requirements and development. These activities include architectural and detailed design and other supporting activities like design review process, evaluating design reuse, version control procedures, design tool adoption. Focusing only on the design efforts for building independent software products like class diagram or user interface, while ignoring other design activities may create complexities during testing phase and maintenance of the system.

C. Just Focusing on Functions

The system should exhibit a whole range of qualities attributes (such as performance, security, integrity, maintainability etc) else it is unlikely to be successful.

D. Lack of Platform Precision

Architects need to be really precise about the specific versions, platform and configurations of each element in order to ensure that you get what you need. This will allow you to avoid the condition where customer can't deploy their system because someone has helpfully upgraded a library for one part of the platform without realizing that it means that something else will no longer work."

E. Technology

Apart from the operating environment, the technology used for designing and implementing today's software systems continues to evolve to provide improved capabilities. As new technologies are continuously emerging, software engineers incorporating them all at the same time whereas in some cases, emerging technologies do not completely replace old ones.

III. BUILDING AN INTUITION ABOUT SOFTWARE ARCHITECTURE

Software developers have institution that different kinds of software architectures, but none have being formalized, or institutionalized. In this section we looked at some of the architectural principles in order to build intuition for the software architecture. Devayne E.Perry and Alexander L.Wolf presented a model of software architecture consisting of 3 components: elements, forms and rationale. Then they developed i) Computing h/w architecture ii) network architecture[3]. We have taken classical architectural discipline of building software architecture like hardware and network architecture because they have conventionally been considered as sources of ideas for software architecture.

A. Hardware Architecture

According to the hardware emphasized there are various approaches to build hardware architecture. For example RISC machines emphasize on the instruction set, Pipelined machines emphasizes on the configuration of hardware components. One of the contrasting feature of this architecture is there are relatively small number of design elements whereas in software architecture there can be large number of design elements. Secondly replication of these elements makes the scale whereas in software architecture adding more design elements makes the scale.

B. Network Architecture

Abstracting the design elements into nodes and connections builds the network architecture. The important feature is to name the kinds of relationships that the two elements hold. The topology it follows can be star, ring, bus etc. One of the important feature of network architecture are it has two elements-nodes and relationships and only few topologies are being considered. However instead of considering only few topologies, we emphasized on several aspects of network topology such as distributed architecture and message passing architecture.

C. Building Architecture

Robert T. Monroe stated that an architectural style provides a design vocabulary (clients and servers, pipes and filters, data and knowledge sources, etc.) along with rules for how elements in that vocabulary can be combined [4].Hence following points needs to be considered for building the architecture:

- Multiple views;
- Architectural styles;
- Style and engineering; and
- Style and materials.

Kruchten's 4+1 View Model takes four views as its starting point [5]. Viewpoint integration is accomplished via a "fifth view" which is a set of scenarios used to validate the other views and their interactions. There are a number of existing approaches utilizing multiple views in software systems architecture. Meszaros explains an architectural view as: "a way of looking at architecture. Each view may have a different concept of elements and relationships" [6].In this paper we are focusing on only implementation view which is the main focus of software developer.

Next is the relationship between architectural style and engineering principles (and, of course, architecture itself). This relationship between software architecture and engineering principles is also of fundamental importance. Lastly, the relationship between materials and architectural style is of critical importance. Certain properties of the materials have been exploited in modeling any particular style of architecture.

IV. CONTEXT AND MODEL FOR ARCHITECTURE

A. Context

Different parts of software product are being characterized according to the entities, properties and relationships. Moreover, our characterization represents a wide variety of evolutionary and development paradigms used in the creation of production software, Following are the considerations:

- Requirements are focused on the information, processing, and the characteristics of that information and processing needed by the user of the system;

- Architecture is focused on the selection of architectural elements, their interactions, and the constraints on those elements.
- Design is focused on the modularization and detailed interfaces of the design elements
- Algorithms and their data types are needed to sustain the architecture and to satisfy the requirements; and
- Implementation is focused on the representations of the algorithms and their data types that satisfy the design, architecture, and requirements.

B. Model

The concept of developing an architecture that we appeal to is that of the standard definition: "The science of building especially designing and building consistent structures" [7].Perhaps more relevant to our needs it can also be defined as: "A unifying or coherent form or structure" [8]. We first presented the model of software architecture followed by architectural styles.
We propose the following model for software architecture:

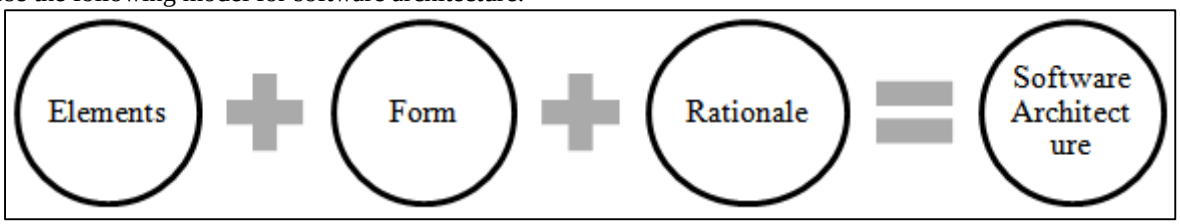

Fig. 3: Software Architectural

Software architecture comprises of architectural elements that have a particular form. We categorize architectural elements into three classes:

- Processing elements: They help in the transformation of the data elements
- Data elements: It contains the information that needs to be transformed or which is being used
- Connecting elements: It can be either processing or data elements. It links different components of the architecture together. Examples of connecting elements can be shared data and messages, procedure calls etc.

Next is the architectural form which consists of weighted properties and relationships. The weighting indicates either the significance of the property or the relationship, or the need of selecting from alternatives.

- Properties are used to identify the constraints on the elements to the degree as desired by the architect. Moreover it defines the minimum desired constraints unless the default constraints are defined by properties.
- Relationships are used to constrain the position of architectural elements i.e. they confine the interaction of different elements and thereby organizing them with respect to each other in the architecture.

Finally the integral, part of architecture is the foundation for the various selections made in defining architecture. The rationale captures the selection of architectural style, elements, and the form. In software architecture, the rationale instead focuses on the satisfaction of the system constraints which are determined by the considerations ranging from various non-functional aspects to various functional aspects such as economics [8], performance and reliability [9].

V. PROPOSED SOLUTION

As we have seen in the paper there is no standard and notation followed in building the software architecture therefore in accordance with these issues and challenges we have proposed different architectural styles which has standard notation of shapes in different styles of architecture. An architectural style categorizes a family of systems that are associated by shared structural and semantic properties. More specifically, styles typically provide four things:

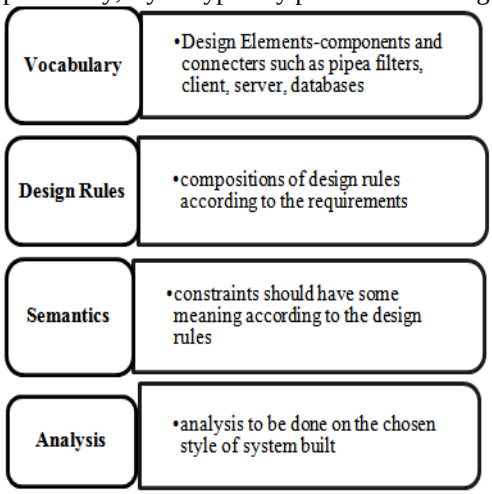

Fig. 4: Properties of Architectural Styles

In this research we focused on following styles of architecture

- Object oriented Architecture
- Pipe and filter
- Layered architecture
- Data centered architecture
- Event based Architecture

The interdependence of processing data elements upon the connections is more obvious: the connecting elements are responsible for moving data around from processor to processor.E. James Whitehead, Jr. Jason E. Robbins Nenad Medvidovic Richard N. Taylor provided the description of the framework of key properties that a software architecture should exhibit on the basis of a component marketplace. He studied existing Software Component Marketplaces: Unix Pipe and Filter , Visual Basic and VBX, chiron 2 architecture[12]

A. Object Oriented Architecture

In this style, data representation and their associated fundamental operations are encapsulated in an object or abstract data type. Instances or objects of abstract data type are the main components of this style. Objects communicate with each other with the help methods and procedure call. It can be multithreaded or single threaded [13].In object oriented architecture the rectangular shapes represent objects and arrows represent method call.

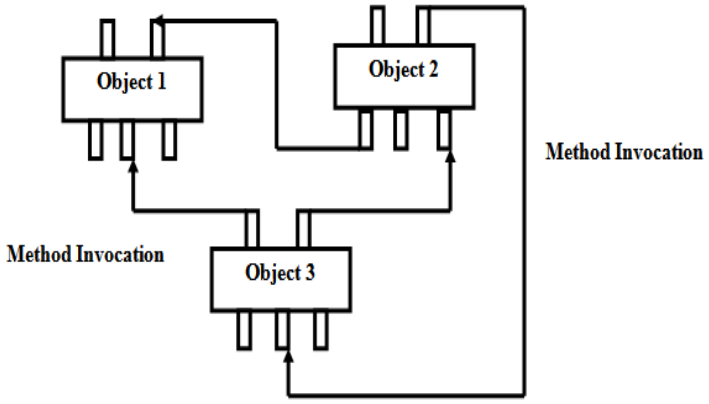

Fig. 5: Object Oriented Architecture

B. Pipe and Filter

This architecture contains set of components and connectors where, components are processing units also known as filters which are linked with the help of connectors (known as pipe) through which data flows from one filter to other. Each filter take data as input which further performs some computation on it and then send it to the other filters for further processing. Filters can be used in sequential or in parallel or in both. Filters are represented by rectangular boxes and arrows represent pipes.

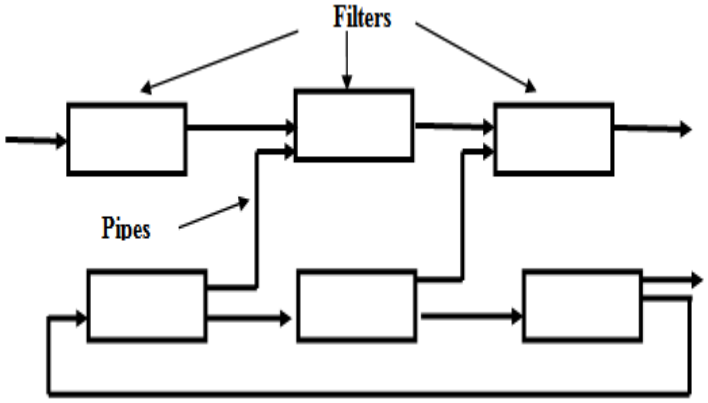

Fig. 6: Pipe and Filter Architecture

C. Layered Architecture

It is a hierarchical organization, each layer acting as a server provide functions to the layer above it and serving as a client the layer below. According to the number of layers, architecture can be 2-tier architecture (having 2 layers only), 3-tier architecture (having 3 layers) and so on. Each layer has its own responsibilities. Two well-known examples of layered architecture are OSI reference and operating system model. In operating system, basically three types of layers are used kernel layer (nearer to

hardware), it provides the services to the utilities (drivers, compilers etc.) to be installed and then application layer which help user to interact with the system.

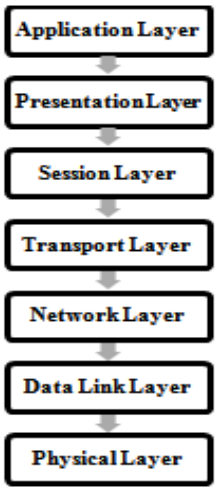

Fig. 7a: OSI Model

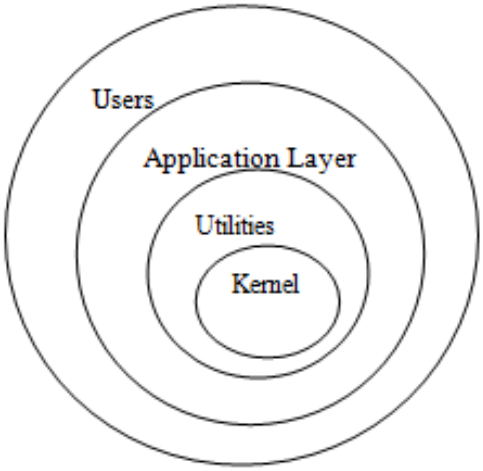

Fig. 7b: Layered Architecture

In OSI reference model, seven layers are used as mentioned in the fig 7a. Physical layer provide its services to the data link layer which further provide its services to the network layer and so on. Layered architecture has mainly two variants- open layered architecture where an upper layer can use the services of any below layer directly and closed layered architecture where an upper layer can use the service of layer directly below it

D. Data cantered Architecture

The main focus is on data. These knowledge experts access the common data in synchronized manner to preserve the consistency of data where transactions on this data are in same order for all the knowledge sources. Data centered architecture is generally used for network based applications and the systems that involve common access to data with loosely coupled components. In this architecture rectangular boxes represent client's system, cylinder shape represents data and arrows depicts data flow.

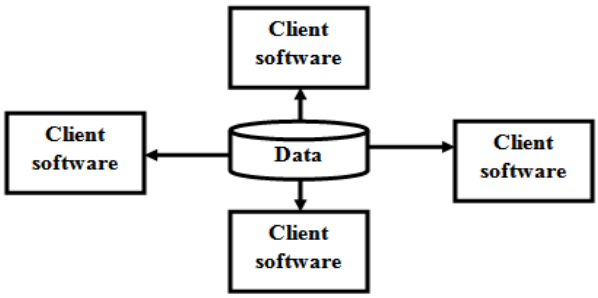

Fig. 8: Data centered Architecture

E. Event Based Architecture

In event based implicit invocation, a module or a component broadcast one or more events instead of invoking a procedure directly while other components can register an interest in an event by associating a procedure with it. Whenever the event is encountered, the system invokes implicitly all the procedures that have been registered their interests for the event.

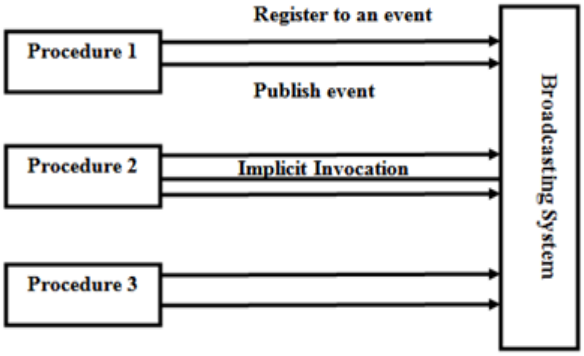

Fig. 9: Event Based Architecture

Interface of component in an implicit invocation style provides both a collection of procedures and set of events. A component can register some of its procedures with events of the system. This will cause procedures to be invoked when their associated events are announced at run time. In this architecture, rectangular box represent procedures and arrows represent method calls.

VI. CONCLUSION

We came to the conclusion that software architecture plays a vital role in IT industry. Establishing an intuition and building the context for software architecture and architectural style is a first and foremost thing to develop architecture. Therefore an architect should start by defining the context of architecture. Then architect formulates a model of software architecture that emphasizes the architectural elements of data, processing, connection, highlighting their relationships and properties and lastly captures the constraints on their realization or satisfaction. Moreover we delineated necessary features of architectural description techniques and their supporting infrastructure. Moreover different shapes follow different standards and notation for various styles of architecture. Architectural style, objects, and design patterns the complementary aspects of design. Software Architecture style has been mentioned more and more in software development today. Architecture style selection is the essential phase in software design because satisfying Quality Attributes is one important issue in software system design that suitable software architecture can fulfill it.

REFERENCES

[1] Mary Shaw, the Coming-of-Age of Software Architecture Research, Institute for Software Research, International Carnegie Mellon University, 2001
[2] Mary Shaw and Paul Clements, The golden age of software architecture, Institute for Software Research International School of Computer Science, February 2006
[3] Devayne E.Perry and Alexander L.Wolf, Foundation for study of software architecture, Department of Computer Science University of Colorado October 1992
[4] Robert T. Monroe, Andrew Kompanek, Ralph , David B. Garlan, Architectural Styles, Design Patterns, and Objects, Carnegie Mellon University, September 1996
[5] Philippe B. Kruchten. The 4+1 view model of architecture. IEEE Software, 28(11):42–50, November 1995
[6] Gerald Meszaros, Software architecture in BNR, In Proceedings of the First International Workshop on Architectures for Software Systems, 1995.
[7] E.J. Chikofsky (Ed), Software Development Computer-aided Software Engineering, Technology Series, IEEE Computer Society Press, 1988.
[8] F.C. Mish, Webster's Ninth New Collegiate Dictionary, Merriam Webster, Springfield, MA, 1983.
[9] B.W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[10] D.E. Perry, The Inscape Environment, Proc. Eleventh Inter. Conf. on Software Engineering, Pittsburgh, PA, IEEE Computer Society Press, May 1989, pp. 2-12.
[11] A Brief Survey of Software Architecture by Rikard Land (Department of Computer Engineering, Mälardalen University, Västerås, Sweden, February 2002)
[12] Software Architecture: Foundation of a Software Component Marketplace by E. James Whitehead, Jr. Jason E. Robbins Nenad Medvidovic Richard N. Taylor (Department of Information and Computer Science University of California, Irvine, 1995)
[13] Software Architecture Styles a Survey, Ashish Kumar, Computer Science and Engineering, February 2014
[14] Mary Shaw and David Garlan, Software Architecture, Prentice Hall of India, 2004.