**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 9

**System Construction**

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Approved By:**

**Academic Year: 2017-18 Semester: I**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction |  |
| 2 | Architecture Objectives |  |
| 3 | System Design Specification |  |

# INTRODUCTION

# *The software engineering community realized that software architecture is not only about structures (components and interfaces), but also about system behavior (interaction between components, protocols). Furthermore, this community introduced an architectural design phase in the system life cycle, in which requirements should be satisfied and which should serve as a basis for detailed design activities. Researchers and engineers in software engineering have adopted the term 'architecture' as well. Nevertheless, there is no consensus about the subject; no universally-accepted definition of the term 'architecture' is agreed upon.*

# *Perry and Wolf (1992) consider a software architecture as a set of architectural elements that have a particular form. Similar to Zachman and Van Waes, they distinguish three different classes of architectural elements: processing, data, and connecting elements. Perry and Wolf consider an architecture as a necessary framework in which requirements are satisfied and which serves as a basis for the design.*

# *Garlan et al. (1995) stated that a system's architectural design is concerned with describing its decomposition into computational elements and their interactions. Design tasks at this level include organizing the system as a composition of components; developing global control structures; selecting protocols for communication, synchronization, and data access; assigning functionality to design elements; physically distributing the components; scaling the system and estimating performance; defining the expected evolutionary paths; and selecting among design alternatives.*

# *Soni et al. (1995) stated that software architecture is concerned with capturing the structures of a system and the relationships among the elements both within and between structures. Software architectures describe how a system is decomposed into components, how these components are interconnected, and how they communicate and interact with each other. Based on a survey on the role of architecture in the design and development of large systems within Siemens, Soni et al. notice that different structures are used at different stages of the development process. Each structure describes the system from a different perspective.*

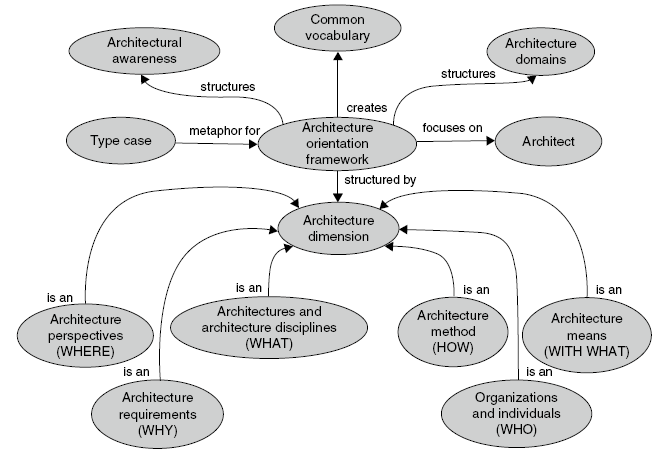
# *Soni et al. argue that the four different architectures they distinguished are needed because of the growing complexity of software throughout history (see Figure 1.3). Initially, only the code architecture was required. The module and execution architecture became necessary when systems became larger and distributed. Now, software engineers would like to use communicating objects and assemblies of reused components. Therefore, a high-level structure is described in the form of a conceptual architecture. On the other hand, Zachman and especially Van Waes reason that their various architectures are wanted as representation for each of the involved actors.*

# *Garlan and Perry (1995) found that the term 'architecture' is used in a number of ways in software engineering. Among the various uses are a) the architecture of a particular system, as in 'the architecture of this system consists of the following three components,' b) an architectural style, as in 'this system adopts a client-server architecture,' and c) the general study of architecture, as in 'the papers in that issue are about architecture.'*

# *A discussion group at Carnegie Mellon University's Software Engineering Institute developed a typical definition: the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time. They represent a spectrum in the software architecture community about the emphasis that should be placed on architecture - its constituent parts, the whole entity, the way it behaves once built, or the process of building it. Taken together, they reflect the various aspects of software architecture.*

# *Software architecture is concerned with the design and implementation of IT systems. From the viewpoint of architectural activity, software architecture covers the steps necessary to design and implement architecture. With regard to the structural aspect of architecture, software architecture describes the structures of IT systems. From this point on, the terms “IT system” and “system” are used synonymously provided no explicit differentiation is necessary. A system is a unit that consists of integrated software and hardware building blocks and exists for the purpose of fulfilling a functional objective. To achieve this objective, it communicates with its environment and must take account of the conditions defined by the environment.*

# *http://www.home.zonnet.nl/azwegers/thesis/figures/2_2.gif*



# ARCHITECTURE OBJECTIVES

* ***To manage complexity****: An architectural model allows one to present the essence of a complex system in a (simple) model. An architectural model supports the ability to comprehend complex systems; it presents them at a level of abstraction at which a system's high-level design can be understood. It supports the analysis of relationships as an aid to understand complexities in a design environment. In particular, an architecture is needed in complex, dynamic environments (Van Waes, 1991). Zachman states that the increased scope of design and levels of complexity of system implementations are forcing the use of architectural models for defining and controlling the interfaces and the integration of the system components (Zachman, 1987). Architectural models abstract away from details instead of from the essential complexity. Brooks claims that 'the complexity of software is an essential property, not an accidental one' (Brooks, 1995; p. 183). Descriptions of a software entity that abstract away its complexity often abstract away its essence.*
* ***To serve as a set of specifications****: An architecture may be seen as a result of the design process. It is laid down in specifications, which are derived from the requirements, and from which the desired system can be built. Specifying an architecture is concerned with the specification of components, their interactions, and the constraints on these entities and their interactions. These unambiguous specifications define the scope of future development activities, and serve as a basis for further design and implementation activities.*
* ***Means of communication****: Furthermore, an architectural model may play the role of a means of communication during a system (re-)design process. The architect can use it to visualise various aspects of the system to be designed, thus providing the various parties concerned with a basis for discussion and decision-making. By producing order in chaos, architectural models help each party to clarify its perception of the problem. Visualisation and explanation of the relevant aspects of the problem area, and the possible relationships between them, supports the various actors to focus their attention on the essential elements, thus providing a basis for discussion of the problems.*
* ***To indicate the most vital system elements****: Furthermore, the architecture determines the nature and quality of a system. As such, an architectural model indicates the invariant or most vital system elements, which must be treated carefully during system re-design. Systems evolve and are adapted to new uses, just as buildings change over time and are adapted to new uses. One frequently accompanying property of evolution is an increasing brittleness of the system, caused by violations of the architecture. Violations of the architecture frequently lead to an increase in problems in the system and contribute to an increasing resistance to change, or at least to changing gracefully.*
* ***Means to reduce the impact of changes****: Another role of an architecture involves its contribution to the effective re-design of a system. The architecture should reduce the impact of changes to the lower component levels, and to as few components as possible. Both for shop floor control systems and for products, it is advantageous to use as many parts of the existing system or product design as possible. In a re-engineering trajectory, an architectural model of the system allows one to pinpoint and discuss the areas requiring major change, and to integrate the new specifications into the existing model. Furthermore, architectural change is not so much determined by the system components, as well by the interfaces between these components; the ease with which components can be modified, replaced, or with which the system can be extended by new components is dependent on the extent to which the interfaces of the new components match those of the old ones.*
* ***Means to gain strategic benefits****: Finally,(product) architecture may have certain strategic importance for a company. The development of a new product brings together a wide range of technologies. Only a few of these technologies contribute to ultimate competitive advantage. Successful companies do not compete on (and even give away) the enabling technologies on which their core utility is based. By the architectural design of functions that can be filled in by cheap, standard components, companies profit from the strong competition in the markets for these components, and are free to focus on their true sources of competitive value. In addition, a company might extend the value of its product by publishing the product's interfaces to the outside world. Other enterprises might use this product as an indispensable part for their own products*

# SYSTEM DESIGN SPECIFICATION

*A modular architecture may naturally result in a layered architecture; modules are assigned to specific layers. Layers reflect design decisions based on allowable relations and interfacing constraints. The layers in an architecture represent allowable interfaces among modules. Modules within a layer can communicate with each other. Modules in different layers can communicate with each other only if their respective layers are adjacent (Soni et al., 1995). A layer builds on its underlying layer, which at its turn builds on its underlying layer as well. Consequently, a layer explicitly uses the functionality of its underlying layer, and implicitly uses the functionality of all layers underneath its underlying layer.*

*Layers are used mainly to solve mapping problems. The mapping task is decomposed in layers: each layer performs a specific part of the mapping. In this sense, the division in layers is part of an architecture. The advantage of layers is the flexibility: changes can be made inside a layer without affecting other layers. A disadvantage of a layered architecture is its rigidity: new layers are hard to be shoved in between existing layers, since this requires a (major) change of interfaces. Examples of the application of layers in mappings are:*

* *the targets of an enterprise must be mapped on its physical processes; therefore, a strategical, tactical, and operational layer are distinguished;*
* *data from a database must be mapped on computer screens; therefore, an internal, conceptual, and external layer are distinguished.*



|  |  |
| --- | --- |
| Layer-1 | Layer Name |
| Purpose | Indicate purpose of the objective here in 3/4/ statements. |
| Related Components | Provide the list of components here |
| Software Interfaces | Indicate the Interfaces used |
| Composition Style | **Generalization/ Aggregation/ Composition** |
| Communication Pattern | **Horizontal/ Vertical** |
| Implementation Steps | 1.  2. |

|  |  |
| --- | --- |
| Layer-2 | Layer Name |
| Purpose | Indicate purpose of the objective here in 3/4/ statements. |
| Related Components | Provide the list of components here |
| Software Interfaces | Indicate the Interfaces used |
| Composition Style | **Generalization/ Aggregation/ Composition** |
| Communication Pattern | **Horizontal/ Vertical** |
| Implementation Steps | 1.  2. |

|  |  |
| --- | --- |
| Layer-3 | Layer Name |
| Purpose | Indicate purpose of the objective here in 3/4/ statements. |
| Related Components | Provide the list of components here |
| Software Interfaces | Indicate the Interfaces used |
| Composition Style | **Generalization/ Aggregation/ Composition** |
| Communication Pattern | **Horizontal/ Vertical** |
| Implementation Steps | 1.  2. |

|  |  |
| --- | --- |
| Layer-4 | Layer Name |
| Purpose | Indicate purpose of the objective here in 3/4/ statements. |
| Related Components | Provide the list of components here |
| Software Interfaces | Indicate the Interfaces used |
| Composition Style | **Generalization/ Aggregation/ Composition** |
| Communication Pattern | **Horizontal/ Vertical** |
| Implementation Steps | 1.  2. |