Providing Reusability in Industrial Control Systems

By

Sheza Javed

NUST2013-62883-MPNEC45013F

Supervised by

Cdr. Dr. Attaullah Y Memon PN

A dissertation submitted to

PAKISTAN NAVY ENGINEERING COLLEGE
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

A thesis submitted in fulfillment of the requirements

For the degree of Master Program

in the

Electrical Engineering (Control)

Department of Electronic and Power Engineering

May 5, 2017

Declaration of Authorship

I, Sheza Javed, undertake that this thesis work titled,

"Providing Reusability in Industrial Control Systems" and the work presented in it are my own. I confirm that:

_ This work has been done mainly while in candidature for a research degree at this institute.

_ In case of consultancy of the publish work of other, I have clearly referred it.

_ All main sources of guidance and help are acknowledged.

Signed:

Date:

Acknowledgement

I would thank Allah Almighty from whom we seek guidance at every stage of life. Writing this thesis has truly been a journey. Not only have I gained knowledge about engineering, but also of myself and a lot of people around me. There have been so many people helping me and supporting me to make this journey a successful reality. I would like to thank them all individually.

First and foremost, I would like to thank my supervisor Cdr Dr Attahullah Y. Memon who truly is my motivator. He has been supporting me and encouraging me at every moment and has given me the confidence to do the best in masters since the day he took my interview on the admission day. It has been a blessing to have a supervisor like him who gives wise you suggestions, responds to you and guides you sincerely. It is because of his vast interests and knowledge about controls that motivated me to work on a different topic like this. I want to thank you Sir, for believing in me and putting so much effort in making this work possible. It would have been nearly impossible for me to achieve this without you.

Moving on, I would be highly thankful to all my GEC members namely, Sir Aleem Mushtaq, Sir Sajjad Zaidi and Sir Naeem Abbas, for their availability, suggestions and guidance at all times for refining my work for its betterment. This has really taken my work to another level. Thank you for giving me your precious time in this busy schedule.

I would thank my family, which includes my Mom and Dad, my brothers Danish and Hassan, my sister Nimra and my husband Talha for always being there for me and not letting me quit at any point!!

Last but not the least, I would like to thank my friends Sidra and Zehra, specially Sidra, for keeping me sane through this whole process and for constantly reminding me of who I am and how far I have come for this.

In the end again, I would like to thank Allah for giving me the strength to accomplish the endeavors that I thought were impossible.

ABSTRACT

Industrial control hardware maybe reused for several purposes. With growth of complexity and software size, there is a need for systems to be flexible and reusable. An interesting question that often arises is whether it is possible to tailor model of a specific control system to meet the requirements of a new one by simply reusing some components of the existing system or we need to build up a completely new system from scratch. Lately, some prominent researchers have shown that designing component based control systems can conveniently provide reusability when the original system is no longer in use. However, providing reusability for control design is not straight forward and poses a challenging problem in absence of a suitable framework.

In this thesis, we propose a framework whereby starting from an existing component based system we measure reusability of various components of the system. Specifically, we formulate a procedure through which we evaluate the system for reusability of its components that may be used for design of a new system. For this purpose we propose a generic methodology namely ***Reusability Design Approach (RDA***). The striking feature of our approach is combination of component based software and/or hardware engineering subsystems to achieve the desired reusability tasks. The proposed methodology is applied to a humanoid robotic hand in order to show the efficacy of our approach. Furthermore, a reusability measurement model is proposed for measurement of reusable components with a reusability index (Sheza Reusability Index, SRI).

Key words: *Industrial Control Systems, Component Based Engineering, Reusability, Reusability Design Approach (RDA), Reusability Measurement, Sheza Reusability Index (SRI)*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem statement:

For decades, industries are working hard to make the best use of the latest technologies to come up with systems which are robust, efficient, smart, interoperable, simple, properly documented etc. to fight with the changes in technologies and improvement in systems. It becomes a difficult task for the design engineers at the back end to build up systems which are low cost, efficient and smart. To reduce their time to market is a difficult task indeed. Object-Oriented (O-O) technology alone is not enough to meet the tough requirements these days. O-O methods focus on developing rich models that would reflect the object of a problem, but this does not necessarily yield the architecture for software to meet the changing in any system. O-O techniques do not particularly deal with independent upgrade of features, most features cannot even be removed, neither can they get replaced, nor can they be used in other applications.

Modern complex control systems of a specific domain do share common and similar functionalities. Designing of control system is essential for an industrial control setup. With growth of complexity and software size, there is a need for systems to be flexible and reusable. Is it possible to tailor the models for any specific control system instead of building up a completely new system from scratch? Different publications have shown that the technique for this is the development of component based systems, which is rather old but not obsolete. Developing a Reusable system is definitely a challenging endeavor.

In this tough growing industrial market, it is very important to lay emphasis over developing systems in a way that they can be made more beneficial. The word ‘Beneficial’ here has a deep meaning. It just does not mean the ability of a system to complete its task, but it is the ability of completing the task with better understanding, better strategy to develop systems which would tell that the engineers to focus over developing systems which can be the foundation of another new system to be developed. A system should be able to donate its parts to other systems as components. Industries at every point need advancement in the older systems to meet the

latest technologies and to improve older network of systems. So there has to be some direction to which engineers can help themselves reduce efforts in developing new systems by reusing their work and make their efforts more beneficial.

The solution of these growing complex systems is a component based architecture which would result in a faster, better and cheaper control system as a whole. The whole idea revolves around reusability concept which is the core objective of this research.

There are a few questions which are to be answered at the end of this work. The questions are:

- Can we reuse existing solutions instead of inventing them?
- Is it possible to tailor the models for any specific control system instead of building up a completely new system?
- What are the conditions for re-using the existing systems?
- Can we build up the new systems using the suggested approach?
- Is reusability a measurable entity?
- What are the pros and cons?

The next chapter describes in detail what reusability means, how it can be possible, what is component based engineering, what are the concepts and architecture of component based software engineering and component based system engineering. What are the differences between these two kids of component based engineering also what is model based development.

1.2 Industrial Control Systems

Industrial control system (ICS) is a general term that encompasses several types of control systems used in industrial production, including supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other smaller control system

configurations such as programmable logic controllers (PLC) often found in the industrial sectors and critical infrastructures. The nature of a general industrial control system is given below.

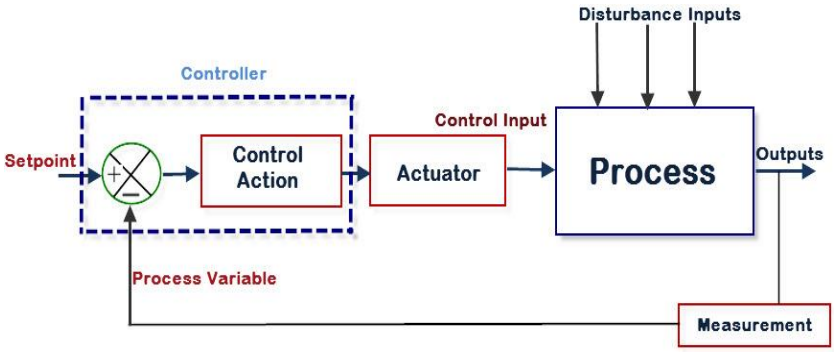


Figure 1: Industrial Control System

Industrial control market is an ever running market for control systems since decades. All the marketers want their product to reach the market as early as possible. This becomes a difficult task for the design engineers at the back end to build up systems which are low cost, efficient and smart. To reduce their time to market is a difficult task indeed. It takes a lot of effort to design that for sure. Every part of the control system has its defined hardware, software and functionalities which run the system. A distributed control network of system is the most emerging organization in the industrial control field. The Distributed control system allows ease in handling the task and consists of autonomous units which operate as of operating entities to provide high operational flexibility and change capability to the system. The control systems which were managed on large and very expensive control architectures have been distributed into sub tasks which are handled by small controllers. Instead of waiting for the master controller to handle all the tasks, we may build up intelligent, small controllers I.e. sensors and actuators to handle the tasks of the system dependently or independently. This will help in providing flexibility to the distributed control components if used somewhere else also. Such intelligent automation elements could be used as common building blocks to compose a manufacturing system. This whole idea is called a fully distributed Control concept. Depending on the functionality and complexity of the control action, industrial control systems are differentiated into several types, but most commonly and widely used control systems are of three types:

Programmable Logic Controllers (PLCs) Distributed Control Systems (DCS) Supervisory control and Data Acquisition (SCADA).

1.3 Component Based Engineering

Component based engineering is a branch of software engineering which focuses on development and maintenance of control systems by modern technology of componentization. Design and development of any control system is a tedious task. Componentization of control system means designing a control system by chopping down the system into functional blocks so that it gets easier to understand and operate the system in the long run. The development of a control systems using component based engineering means designing, developing and evaluating components which can be called off the shelf components to be used in other systems easily, save time and money when building large complex systems, reduce software hassle, deliver better quality systems to the market and reduce the number of faults or errors expected in the large complex systems.

The concept basically targets the idea of a flexible reusable system development. In precise words is it to say, "Divide and conquer ".

There are two types of component based engineering like Component based software engineering (CBSE) and Component based system engineering (CBSysE).

1.3.1 Component based Software Engineering

The component based software engineering (CBSE) is an engineering in which software is code is generated, specified, designed, implemented and most importantly maintained in the form structures which can be reused. Object Oriented (O-O) architecture focuses on developing rich models that would reflect the object of a problem, but this does not necessarily yield the architecture for software to meet the changing in any system. So, the CBSE uses the principles of software engineering to apply same idea as O-O to generate software systems. The main focus is to design reusable and adaptable existing components and systems not just coding in a particular

way. CBSE is a paradigm that aims to provide the platform for designing and maintaining pre-defined software compositions to be reused, instead of just programming. This, in theory, allows the software systems to be easily assembled, programmed and less costly to build. This statement varies from system to system of course. There is a ratio and proportion to every system to be designed in this way. But in general, the systems to be built on CBSE would not only be simple and cheaper, but adaptable, robust, and updatable. This in turn leads to a better quality and better standard of the generated systems.

1.3.2 Component based Systems Engineering

The development of Component based systems engineering (CBSysE) emerged from the Component based software engineering (CBSE). Systems engineering approach is taken into consideration to develop the idea of Component based Systems engineering (CBSysE).

A system may be defined as a series of interrelated sets or components joined together to work for a common task. As the control theory comprises of very complex systems there is a definite need of engineers to handle those systems in a systematic manner. The complexity of systems is as a result of interconnections, interdependencies, interrelated components of a system or the systems themselves. Because of this complexity, a system may not be engineered independently as it has to be interactive to form a working system. Therefore, engineers are supposed to look into all these paradigms and conditions and then come up with smart systems which can be a help to the industries. So, it is not just about a component to work efficiently, it is about a complete system as a whole. The components must be engineered within the context of their place within the system. This approach is called systems engineering.

CBSysE, as the name suggests, is the engineering of systems using the component based approach. This will be discussed in detail in the coming chapters.

1.4 Reusability

What is reusability? The literal meaning of this word is “to use again”. In Control theory, it is more than that. I.e. it is the ability of code or a design or a system to be usable, to be transformable, to be recyclable, to be reclaimable and to be utilized and reused to an extent in another application or system in the best possible way. Reusability is the degree to which a component can be reused. It reduces the software development cost and sometimes complexity by reducing the amount of coding and enhancing the system integration. There are a lot of factors which affect reusability and the extent to which reusability can be done, also factors which can measure the reusability of a system. The components have several factors to be considered while we conclude how reusable a system is. At a higher level, reusability can be broken down into two aspects, i.e. usability and usefulness. [1].

Reusability= Usability +Usefulness

As per definition, *“Usability is the degree to which software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.”* [A]

It is an extent to which component is ‘easy’ to use with respect to the amount of effort required to use that component. Whereas usefulness is the ‘frequency’ of suitability for use. Usefulness depends on the functionality and quality of a component.

In this tough growing industrial market, it is very important to lay emphasis over developing systems in a way that they can be more beneficial. Beneficial here has a deep meaning. It just does not mean the ability of a system to complete its task, but a better understanding would tell that the engineers should be focusing over developing systems which can be the foundation of another new system to be developed. A system should be able to donate its parts to other systems as components. Industries at every point need advancement in the older systems to meet the latest technologies and to improve older network of systems. So there has to be some direction to which engineers can help themselves reduce efforts in developing new systems by reusing their work. This gives a short description of reusability.

1.5 Fuzzy Logic and Reusability Measurement

Fuzzy logic is a methodology which is used to solve complex vague statements and then make decisions out of it. The main idea behind fuzzy logic is that entities in the real world do not fit into fixed net categories. So we need to figure out ways in such scenarios. For example deciding the amount of tip in a restaurant cannot be fixed without considering factors which would affect it. It cannot always be the same amount. It depends upon the service and quality of food. For that, we need to define some rules to figure that out. Figure shows a Fuzzy logic system that takes vague statements and data to make decisions.

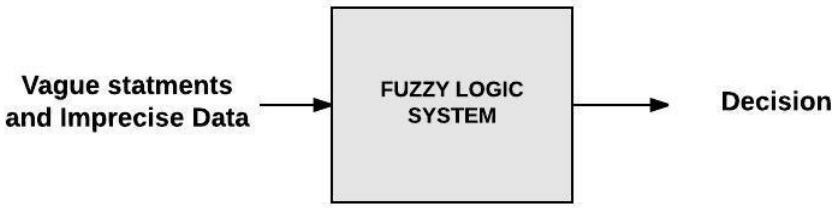


Figure 2 : Fuzzy Logic

In the same way in our case, reusability is an entity which cannot be measured directly, so we have designed a fuzzy model to measure it. The idea of reusing software components has been present in software engineering for several decades; component reuse is still facing numerous challenges.

1.6 Proposed Idea and Thesis Outline

This thesis proposes the path of finding and describing ways in which industrial controls can be built in a way that they can be reused partially/fully for other similar/most similar applications. The approach is to build the systems using the new approach coined as Reusable Design Approach – The RDA. This is explained well in the coming chapters. This is done by Component based software engineering (CBSE) and also component based systems engineering (CBSysE) approach to provide bright and wide perspective of looking at existing industrial control systems and design systems so that they can facilitate reuse. To raise the level of abstraction for future designs and the frequently used functions to save time and money, i.e. provide an economical solution to the demanding future industries. This may lead to reduce the cost considerably.

Chapter 1 is the introduction to industrial controls, component based engineering and reusability. Chapter 2 describes in detail what component based engineering is. What a component and its characteristics are. Also, it explains in detail about CBSE and CBSysE. Then chapter 3 tells what component based development for reuse is. It identifies the characteristics of the suggested methodology called The RDA. Moving on, it explains the systems engineering approach for reuse and the benefits of the suggested approach. Also we continue to give a small example showing the reusability in an industrial control system with conditions applied to it.

Chapter 4 explains how reusability can be measured. Chapter 5 gives the conclusion and future recommendations.

CHAPTER 2

COMPONENT BASED ENGINEERING

Component based engineering is a branch of software engineering which focuses on development and maintenance of control systems by modern technology of componentization.

Componentization of control system means designing a control system by chopping down the system into functional blocks so that it gets easier to understand and operate the system in the long run. Component based engineering has categories, as mentioned in the first chapter, namely, Component based software engineering (CBSE) and Component based system engineering (CBSyE). The concept basically targets the idea of a flexible reusable system development. In precise words is it to say, "Divide, Conquer and Rule".

This chapter tells in detail what CBSE is. Also it describes the whole idea of Component based system engineering. Then moving on to explain the reason why and how sometimes the component based system engineering and component based software engineering together can be a better approach.

2.1 Component

The word component literally means an element or part of a larger whole. In engineering an ideal component is a self-contained subsystem which can be used as a building block in the design for large systems. A component can have a complex internal structure, which is of no concern and is not visible to the user but is a useful entity for the system engineer. An example of component would be automobile or a heat furnace at home. A component is a set of instructions to implement a task. What a component contains all depends on the type of control system and its requirements. A component is a modular, portable, replaceable and reusable set of properly defined functions.

"A component is a unit of composition with contractually specified interface and context dependencies … A component can deploy independently and is subject to composition".

"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only..... A software component is independently deployable and subject to composition by third parties", [Clemens Szyperski].

The most important property a component is supposed to have a good communication and compatibility network. Secondly should be able to do multitasking. A control system which is component based should have control functionality with respect to the component framework I.e. it should be applicable with minor adjustments into the new system. This means that a component which is tried and tested can be used into another similar or partially similar application with some adjustments or no adjustments at all. If all the components are connected and interlinked, they do have the authority to respond to any sort of changes in the systems which are governed by the respective controllers.

When building up a system, it is important for the organizations to lay emphasis on reusability of a component regardless of what the future requirements of the component will be. As mentioned before, extra effort will be paid for any additional functionality if demanded from the already built component for a particular system to make it more useful. One of the biggest properties of a component which makes it reusable is that it can be built as an independently executable entity, an independent block which makes it best suitable for reuse into other applications as well. Component may be a combination of one or more executable objects. Components vary in sizes. Say, a component may be a complete system, or a small part of that system, it can also be a combination of software chopped into different tasks.

Today components are seen a step beyond objects. Components use specified interfaces just like objects to plug-in to a system. But components are ahead of objects because of their better addressing the need for replace-ability and enabling reuse at a higher level making complex system designing an easier objective to achieve. At a larger scale, components sometimes also take majority chunk of the system to be easily plugged-in like a reusable flexible unit. Due to the fact that component comprises of a variety of commands in just one block, their size is such that it would make the life of a design engineer easier while constructing a system. This, in the long run, also makes the tedious task of assembling and integrating components easier as it reduces the number of parts needed for construction of a system.

Examples of component in different industrial systems would be:

- A conveyor belt control (full system)
- A robot manipulator wrist
- Arm control of a robot
- A temperature control component
- Drive controller
- Control functionality of PLC
- Attitude control of a space craft
- Attitude control subsystems: RWA and RCS
- Propulsion subsystem for a SPHERES controller
- Generic component for character recognition

Table 1: Example of Components

2.2 Characteristics of a Component

There is variety of categories for a component to be divided into. A few examples are explained here. Like for example, commercial off the shelf components which are readily available in the market [12]. Then there is a category called qualified components in which the software engineer has to make sure that not only the functionality but most importantly they have to assure the reliability, reusability, usability requirements for the system to be built with them. Another type is called the Adapted components which are adapted to be modified for a system. Updated components are the components which are used where ever there is a chance of update in the system, so these components can be replaced for new versions.

A component may have the following characteristics:

- A component is an independent entity which is executable.
- It can be used in unpredictable number of combinations irrespective of the reason of its development.
- Components can be combined with other components of the same family.
- Components are properly explained, and have a defined interface in which they can be plugged in.
- A component is implemented by using its specified code and defined tools.
- It is a subsystem which is self-contained, reusable piece of software which is independent.
- It can be plugged-in to different applications where suitable with relatively less efforts.
- A component is composed of complete set of instructions for a particular task.
- A component is a complete set of code which can be plugged into other components with very little effort, which makes it reusable.
- If a component is to be sold, it is made sure that it is compatible with the other system as well.

2.3 Component Based Development

Component based development (CBD), also known as the development with reuse, deals with development of reusable entities. CBD provides a platform for development of components when needed. In Component based software engineering (CBSE), it is desirable to reuse the existing components instead of developing them. This actually reduced both cost and time to market, but it requires complete knowledge and research to be implemented. Therefore, there has to be something that provides a solution to the existing systems as well, ways of plug-in components into systems for fast and better quality systems. Not just the development of components, but their compatibility with the system is very important to notice.

A lot of systems contain similar or identical things which engineers develop from scratch for a new system, which is more expensive and time consuming. To meet this challenge of fast and cheaper system applications, the approach called CBD is introduced. CBD can be explained by saying,

“Reuse, do not reinvent” and,

“Rearrange the prebuilt components instead of coding line by line again”.

The importance of Component Based development lies in its efficiency. It takes only a few minutes to assemble the stereo system because the components are designed to be integrated with ease. CBSE encourages the use of predictable architectural patterns and standard software infrastructure, thereby leading to a higher-quality result. Therefore, component based systems development is said to be a reusable flexible approach of building or designing systems in industries.

2.4 Component based software engineering

Component based software engineering (CBSE), is the domain of engineering in which the software code is specifically generated for particular control system. “CBSE is a process that aims to construct and design software systems using reusable software components”. The code is designed, implemented, tested and then maintained through the predefined communication interfaces. CBSE has been an interesting topic for the software engineers in industrial development. If one is familiar with Object Oriented Programming (OOP) approach, CBSE gets easier to understand. OOP can be reused in the form of objects. A programmer can create a new program and inherit the objects into the new program. If not managed properly, this can create issues. But if a complete component, which is tried and tested, is added to the new systems this will make life much easier. This will cut a lot of hard work with establishing the usefulness of the tested components. There will be for sure some testing required for the pretested components according to the new program/system designed. This is much of a reuse based approach. CBSE enables reuse, but reuse alone is not sufficient for CBSE. The motivation is productivity, quality and the maintenance of software controls. If reuse is done for a system, means the component development would reduce and system can build faster and this would reduce time to market. This is not guaranteed though, because there are conditions that apply from situation to situation.

CBSE allows the use of predictable architectural patterns which intern result in a more robust, reliable, and higher quality of system design.
There are different aspects of looking at the implementation of CBSE. For instance, developing a component based system from scratch is the best suitable example. Then, componentization of a few parts of the system which are possible can also be a scenario. Connecting a component based software system to the hardware in the industries is a big challenge. It brings in compatibility issues in as well. Generating a universal control system to handle the whole system of different tasks, instead of having a lot of controllers in the same system is again a valid example of component based engineering.

CBSE embodies the “reuse, don’t build” philosophy.

Andreas Speak in his paper “Reusable industrial control system” specifies a few techniques which would help and support the communication interfaces of the reusable components with the hardware in industries. There were few requirements to facilitate the idea which were integrated control functionality, multitasking control and platform independence. He aims to explain the solution techniques namely, object oriented architecture, architectural pattern, control framework, Architectural model for component based frameworks. These techniques are introduced to facilitate and support reuse on different level suitable to specific problem to be solved by the control system with communication of hardware with the software components.

But this is out of our research for this thesis. This thesis focuses on generating a platform for engineers to ponder over before generating any control system to maximize the quality of their hard work and effort. We limit our work on designing and developing component based systems and their conditions.

2.5 Reusability of a Software Component

Applying reusability concept is a specialized task due to the complexity involved in the development of a system nowadays. The complexity is increasing to levels that if older

developed systems can help building up the new systems, much development and design time can be reduces including reduction in effort as well. Reduction of effort of building up a system through reusable components is still questionable, because it depends on the architecture of the system and its requirements for how much reusability can help. It shouldn't be a problem if the components at the first place are built properly keeping in mind the characteristics to make them easily reusable. The main idea of reuse is, to use previous data/components to build new applications or systems.

Reuse can be of three types, ***application system reuse, component reuse or function reuse*** as **s**hown in figure below [14].

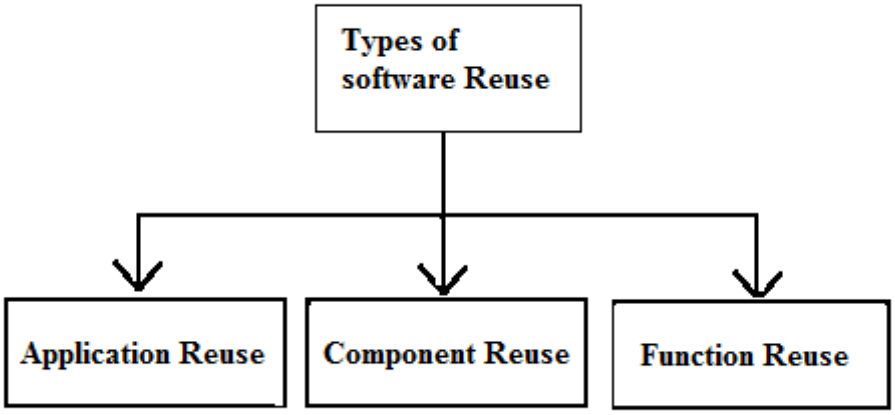


Figure 3 : Types of Software Reuse

Application system reuse is a reuse in which the whole application can be incorporated into the new system. The component reuse can be described as a type of reuse in which subsystems of the application can be reused into another application where needed. The third and last type of reuse, which is function reuse, is the type of reuse in which software components can execute one single well defined function.

The main objective of reusable software components is to minimize repetition of work, the complexity of system design, development time, time to market, quality of the product,

reliability of the system, improve reusability and portability of the systems to make life easier for industries.

Component development is a tedious and tough task for engineers. It requires effort to jot down all details related to a component to be developed. Engineers establish architecture for the system and the whole team determines how a developed component can serve as commercial off the shelf component (COTS). COTS component is a component which is easily available and can be plugged-in to the systems without any changes. Other than COTS, a reusable component should go through three phase processes which are;

- Component qualification; components are identified as per requirements of consumers
- Component adaption; components are modified to meet architectural requirements
- Component Composition; the architecture depicts the end product from the nature of connection of that component

2.6 System Engineering

What are systems? Systems are a series of interrelated components or parts which work together to produce result for a common purpose. With the passage of time, the systems developed today have become more and more complex which is due to interconnections, interdependencies, and communication between the components or the combination of components which form a system itself. So, components are supposed to be designed and engineered keeping in mind the interconnections and behavior between the components for the system. The components should be engineered within the context of their place in the system and its requirement. This approach and idea is called systems engineering.

Apart from this, there have been loads of definitions of systems engineering before. In the International Council on Systems Engineering:

"Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the

development cycle, -documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem:

- Operations
- Performance
- Test
- Manufacturing
- Cost & Schedule
- Training & Support
- Disposal

Systems engineering integrates all the disciplines and specialty groups into team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs. " *[[10] International Council on Systems Engineering. "What is Systems Engineering?" Online.*

2 June 2003.]

Another definition comes from the NASA Systems Engineering Handbook defines systems engineering as "a robust approach to the design, creation and operation of systems" . *[[19] National Aeronautics and Space Administration. NASA Systems Engineering Handbook. June 1995]* . This approach focuses on alternative design concepts, defining the system goals , selecting and implementing the best design after verification and implementation. Finally assess how the system meets the defined goals in the end.

Moving on to another definition from MIT, the MIT's engineering systems characterizes the system engineering as " a process for designing systems that begins with requirements, that uses and/or modifies an architecture, accomplishes function and/or physical decomposition, and accounts for the achievement of the requirements by assigning them to entities and maintaining oversight on the design and integration of those entities".

[[16] Massachusetts Institute of Technology. The ESD Symposium Committee. ESD Terms and Definitions. Version 12. 19 Oct 2001]

There are many definitions of systems engineering but a few have been quotes above. The definitions mentioned above have almost the same idea of systems engineering. The systems are designed to have a lot of sub systems which contain everything from scratch to the completion of

the whole system, i.e. from electronics to the product itself. In simple words, the System engineering also process inclusive. It involves evaluation of the customer needs and requirements at each stage of the development lifecycle. It is important for the engineers to have complete knowledge of not only their domain or the subsystem they design, but also the interactions of the subsystems and how it affects the environment.

Caldwell and Chau note that, "Each person must be able to see a larger portion of the whole than the traditional partitioning according to subsystems" [A]. In particular, an avionics engineer must not only understand all of avionics but also their many interactions with other parts of the space system [A]. That is how the system engineers can lower down the risk and cost of alternative designs with evaluation of systems at the development level.

2.6.1 Component Based Systems Engineering

Component based system engineering (CBSysE) is somewhat closer to component based software engineering. This is because these engineering domains talk about reusability of the components and its frameworks. The component based engineering revolves around the idea of utilization of the control system parts in the maximum possible way. For this growing industrial sector and complex controls, reuse is surely and clearly a partial solution but it really plays an important role even if it helps a bit. It’s cost effective and provides a better understanding to the complex control systems and its applications. Systems engineering, as explained in the previous subsection, is responsible for the design and specifications of the control systems.

Reusability using component based approach can be implemented on the design level. This is the basic point to focus for component system engineering. It means that when the system engineers are designing a particular system, knowing all the design specifications, all the assumptions, the complete model which should include properly and thoroughly documented design rationale. They can actually work on finding the reusable parts of that system. As far as the component based approach is concerned in terms of software there are pros and cons in every method. Where software reusability

has many advantages, it has for sure provided a new platform to the software engineers regarding reusability, it has also resulted in spectacular losses and limited success in some scenarios. This is where the approach to reusability is done by component based system engineering which involves designing reusability at design and development level. For example in spacecraft controls NASA has lost billions of dollars and some scientific missions due to poorly designed software techniques. Again, when software reuse has given so many benefits to the growing industries, it has not benefited the space industry much.

Reusability is done at software level before but control engineers must go back at an earlier development phase of the system and then apply a more beneficial reuse. This idea would result in reuse without some draw backs inspected before [2]. In the model driven approach and component based system engineering the task is to find reuse at the design level before generating the software for a particular system. For this purpose the system engineers after a thorough review of the system requirements, design specifications should look up for reusable components at the design and development level. Changes at this level are reviewed or made by system engineers and the experts who are more likely to understand the application's engineering requirement in depth and less likely to make change that violate the basic engineering assumption underlying. Although the systems are developed using the component based approach, but if the systems engineering principles are applied to it, it helps the engineers to analyze interconnections of the software at every stage of the development lifecycle. Requirements specifications are analyzed even before any hardware implementation is done or even before the code is written. Testing and recognition at the early development stage decreases the risks of any mistake and thereby reducing cost of correcting the mistakes.

The first step in developing a system engineering component based approach is the decomposition of the system. It all depends on the properties and type of the system to be working on; there may be functional, logical or physical decomposition. For example, functional decomposition is a natural approach for designing of a control like spacecraft. In this functional decomposition, the components are grouped into the subsystems based on functionality which they provide to the system as a whole. Like

every complex system, there is supposed to be a controller which integrates all the subsystems and allocates the tasks to them. The second step is the decomposition of the subsystems into their constituent subsystems. The decomposition ends when we reach the hardware elements of the system. For instance we talk about the functional decomposition of an industrial system, it might yield three levels of decomposition namely the controller, subsystems and individual components. Let us take an example of a spacecraft and its decomposition. The figure below shows the decomposition of a generic spacecraft [6].

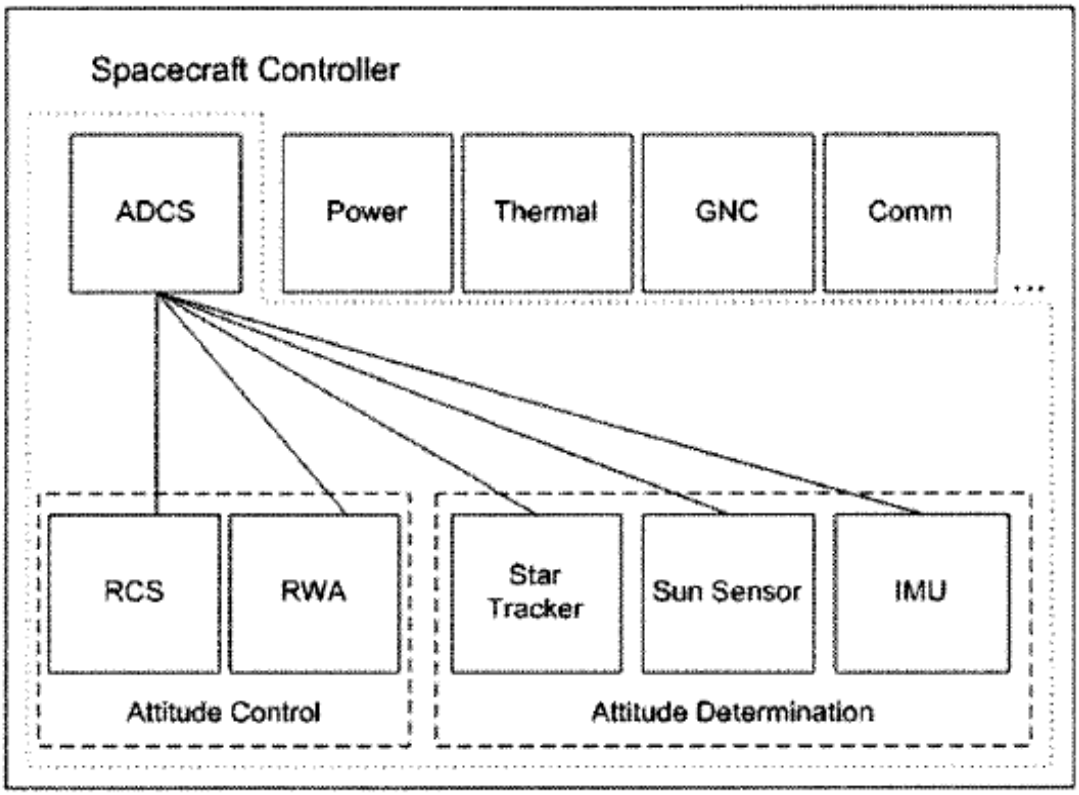


Figure 4 : Decomposition of a spacecraft controller

After the system is chopped down into its components, the next step is to develop the system using component based systems engineering approach, using individual components moving towards the controller of the system. Because there are a lot of systems which have the common development patterns as given by different companies,

it is possible to build a generic model for a given system which makes it reusable. Like the component based software engineering, this approach also suggests building libraries of the components. This as mentioned before as well, makes it easier to develop new systems instead of making them from scratch. The subsystems are defined and refined in detail to reflect a specific goal of a project. They are then combined to form a system using the generic models which allows this technique suitable for a wide range of applications.

The foundation of this thesis has been laid through the two chapters explained in detail above. We are now heading towards the idea suggested in this thesis. The next chapter describes what a reusable component and its properties are. It also explains about reusability of software systems and its components. How do we make systems engineering reusability possible? In addition to all the work, to make it easy for the readers, we present a simple generic example and also show results showing reusability using simple fuzzy logic in MATLAB after defining some reusability metrics.

CHAPTER 3

PROPOSED COMPONENT DEVELOPMENT TO PROVIDE REUSABILITY

The main objective of development of a reusable architecture is to minimize repetition of work, the complexity of system design, development time, time to market, quality of the product, reliability of the system, improve reusability and portability of the systems to make life easier for industries. *The two main types of software reuse development.* One is the development of system *with reuse* and the other one is development of components *for reuse*. In simple words, there are two types of software reuse, with no change to an existing component, and with change. In the first case, which is development of system with reuse, applications should be developed by using the already tried and tested components available for the design. This will eventually reduce the time and cost of the new system to be developed, in short enhancing the quality as well. The second type as mentioned above is development of components for reuse. In this approach the components should be built keeping in mind maximum possible reusability. The components developed should be able to fit into a system in the best possible way, should be compatible with a variety of applications of different platforms. The figure below shows the steps that should be taken to develop a reusable architecture.

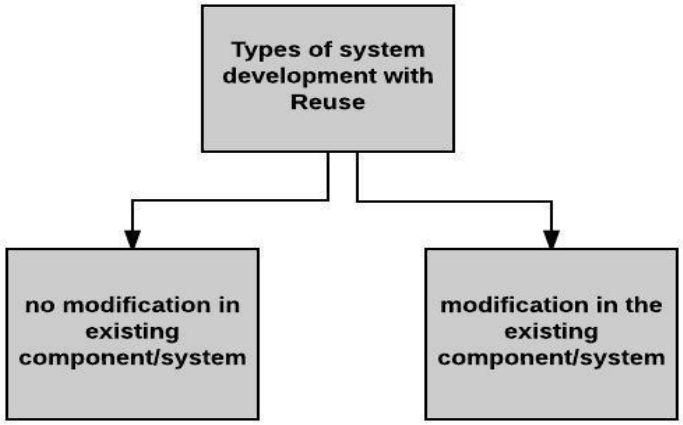


Figure 5 : Development for Reuse

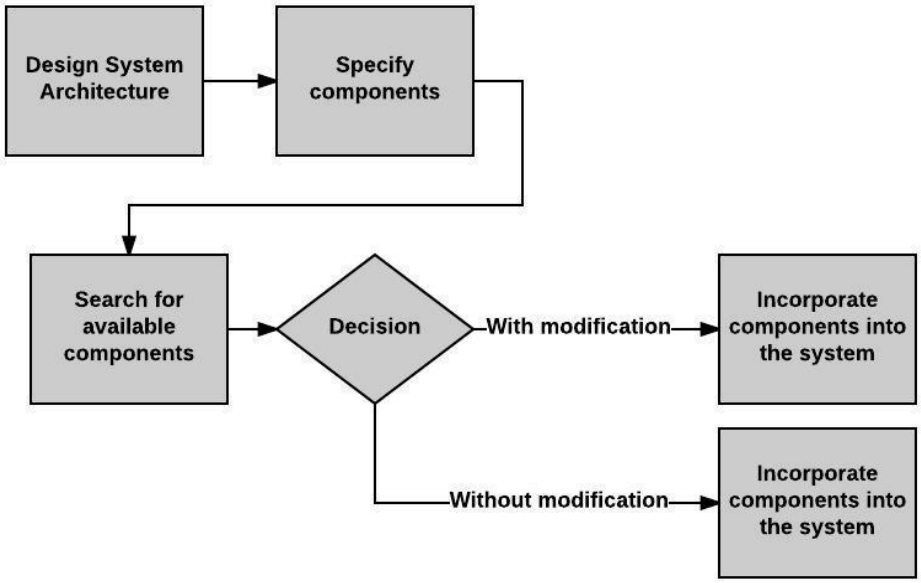


Figure 6 : Steps to be taken in development of Component Based System

This chapter now moves into what a reusable component is, and what is component based development for the purpose of reusability. This work doesn't target a particular industrial control system, but continues to explain the idea generically. After the detailed explanation of reusable component development, the characteristics of a reusable software component are mentioned. The question is;

How do we prove if the whole story can be practical?

Can we measure this vague idea of reusability?

Are here any metrics to measure reusability of a component?

Let's have a look.

3.1 Reusable component

The software reusability and component based development has brought new demands on the quality of software. There is hype in the organizations to build the most useful system along with perks. When components are to be transferred between organizations the demands are even higher. If a component does not meet the requirements of any system, it will die its own death. The components are supposed to be tested thoroughly and more attention is to be given to components than simple programs, because the components are to be plugged-in to other systems. That means one error in a component ends up with errors wherever it is used. If the project is of a higher level, than the reputation of the project is at stake.

Therefore, a reusable component should be deigned in a way that it serves and fits in the required behavior in the intended context. For a component to be built for a higher level project then various factors are to be considered for reusability. That component might be used by the developers for different environments like technical etc.

There are some properties that generally a reusable component may possesses:

Reusable Component Properties	Description
Well documented	A properly documented component would make it easier for the user to understand the properties of that component which attracts component buyers or the users. It reduces the hassle of knowing the functionality and credibility of the component. This assures the quality of the component to the user which is a fundamental factor for them. One way or the other, this is a necessity to make sure the user is confident enough to trust the work. The documentation description must haves include the general information, functional description (detailed information), an acceptance test to show the component works well, and extra additional description like compatibility, support, any unsolved bugs or issues for that component.
Independent	A reusable component has to be an independent entity. If it is dependent, component reuse is difficult to implement. An independent component can easily be plugged into other applications.
Interoperable	Interoperable means, a reusable component should be able to interact with other components. It is a significant property because interaction is required when time comes.
Compatible	If components are not compatible with other systems, it's a hurdle because then it is of no use for other different software. Compatibility ensures maximum reuse.

Table 2: Properties of Reusable Components

The requirement of any reusable component depends entirely on the architecture in which it has to be used. The requirements and characteristics of a reusable software component are defined in the next sub heading.

3.2 THE REUSABILITY DESIGN APPROACH- RDA

Designing the complete Industrial control system and its simulations is beyond the scope of this thesis. So, our focus is not the modeling of a system or designing the control for it, but it is something ahead of it. This work targets on generating a generic a platform to look forward to reusability of the control systems. For that, we have designed a Reusability Design Approach called the RDA.

As mentioned under the heading of component based systems engineering (CBSysE) in chapter 2, it specifies to design the whole system and its requirements before the code is even written so that reuse can be made more useful and beneficial. The system engineer is to develop the whole idea for a particular industry just leaving the code behind for implementation. Changes at that level can be made (or at least reviewed) by system engineers and domain experts who are more likely to understand the application's engineering requirements in depth and less likely to make changes that violate the basic engineering assumptions underlying the system. This helps reduce the chance of any error before the complex part of writing down the code even begins. We tend to merge the component based software engineering and component based systems engineering to form the RDA

Reusable software component is characterized in to two parts; *Formal specifications and Informal specifications*.

The formal specifications are the technical software related specifications whereas the informal specifications are non-technical. The informal specifications are to be treated as the systems engineering approach for component reuse. The informal specifications give a reflection of intent specification designed by Nancy G. Leveson for improved software reusability [4]. So, before the formal specifications, informal specifications should be written.

Formal specifications are specified keeping in mind technical aspect of the component design. The formal specifications are further classified into *internal* and *external* specifications. It is the internal and external specifications which are considered while retrieving the components. The feasibility of these components for any particular project is determined by formal specifications.

3.2.1 Formal specifications (Technical Specs.)

1. Internal specifications

Internal specifications state the internal aspects of the component. They deal with the language, architecture, and design descriptions of a component. The internal specification also tells the type of the component, i.e. its true nature, say, it is an executable component, or a code component, or a design component. An executable component is a component whose code is not accessible and it is just executed. A code component is a piece of component which is used in designing other application systems. Similarly a design component can be used in building up systems wherever required. The design component is a complete set of instruction which may have a couple of code components as well. The architectural component comprises of the internal structure of a component, the involvement of different elements in a component and their relation. In other words, this component shows the blueprint of the components and its elements along with the collaborations between them. Encapsulation is another property which falls under the formal internal characteristics of a component. Due to such complex systems it becomes necessary for the design engineers to hide the important credentials of that system for security purpose. Encapsulation can be performed on different parts of a project/system, for example, on a design description etc.

These generic formal component characteristics can help the design engineers in developing a new control system application in different programming languages.

- **Language**: *On which the component development has to be done.*
- **Type:** *Design component, Code component, Exécutable component.*
- **Design description:** *Design details*

- **Architectural aspect:** *The architecture of component. The internal structure of a component, the involvement of different elements in a component and their relation. In other words, this component shows the blueprint of the components and its elements along with the collaborations between them.*
- **Encapsulation:** *Encapsulation is to hide the important credentials of that system for security purpose.*

2. External Specifications

External specifications should deal with the external aspects of the component. It tells how the component will react with each other or with other applications. There are some properties of the external specifications which help understand what they actually are.

Function is one of the properties which fall into the category of external specification. A component is characterized by the function it performs. There are a lot of functions which a component may perform according to the requirement of the system, but may or may not affect by other functions. The functions which are affected by the other functions are to be called active functions, whereas the functions which are not affected by other functions are to be called passive functions. Another external feature is the technological aspect of the component. This should specify on what platform it was built on and possibilities of that component to be used in other applications, I.e. compatibility.

Another feature is interoperability. Interoperable means, a reusable component should be able to interact with other components. It is a significant property because interaction between the components may be a requirement of a system designed. Interoperability also helps components to be less interdependent but they are able to interact smartly. External characteristics also define the portability of a component. For if the component is portable or not? This means, that if a component is built on a framework it should be clear what framework it is built on, if it can be ported to another framework or not.

- **Function:** *Function of the component. Includes description of all functionalities and effects of one function on another.*
- **Technology:** *Compatibility of technology with other frameworks.*
- **Interoperability:** *Whether interaction is possible within component or between components.*
- **Portability:** *Current platform of component. Also if it can be shifted to any other platform.*
- **Integration Framework:** *Framework helps components to integrate with each other easily.*

3.2.2 Informal Specifications (Non-Technical Specs.)

Unlike formal specifications, informal specifications have no concern with the technical aspects of the reusable component. It enlists the features of the reusable component that deals with the non-technical properties so that similar or most similar reusable components can be retrieved without getting into functional details of the components. Some of the informal specifications include the context, i.e. the details of where and in which scenario the components can be used or reused, the level of reuse is another property that helps determining at what level of the system a reusable component can be plugged into. The intent of reuse has to be specified to define the objective of the reuse of the component, means, what is the target problem solving of that component. This also helps in defining the agility of the component which is another informal specification. Agility of a reusable component is to be seen as the age of the component, i.e. from the instance the component was created to the time it is still considered to be used in a system. If a component is too old, it may not be updated to the latest trends and technology, so it reduces the intent of reuse in applications.

The last and final feature that falls into the category of informal specification of a reusable component is the source of component availability, i.e. how and where are these reusable

components available? Is there a framework for a particular language developed? Are these available as source codes online? What is the compatibility of the available components?

- **Context:** *Portable situations in which the components can be reused into, assumptions constraints, extra details of system. Portable target domains are to be defined.*
- **Level of Reuse**: *At what level or phase of the development lifecycle a reusable component is plugged.*
- **Agility:** *Defines the age of the component. More it is used in applications, more it is stable and reliable.*
- **Intent of reuse:** *Defines the objective of the reuse of the component.*
- **Possible errors (safer reuse):**

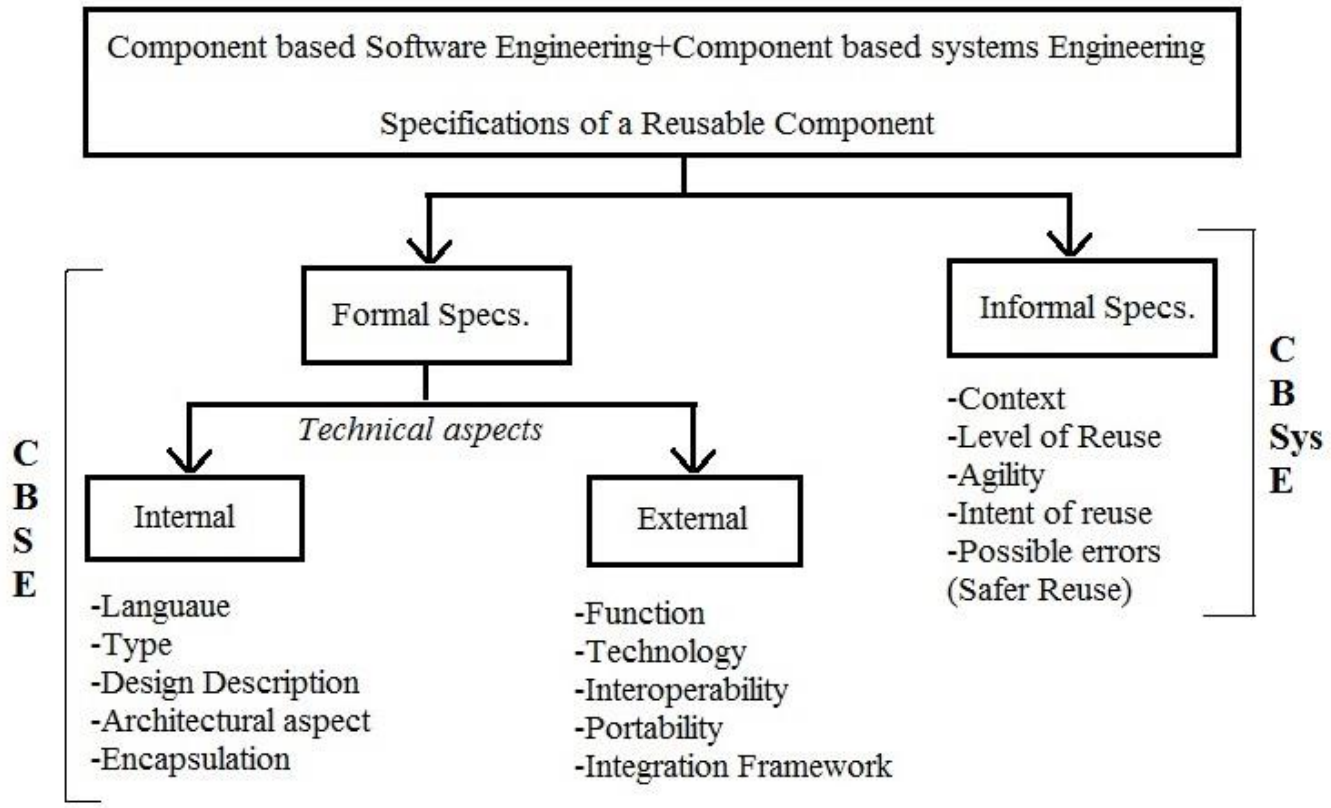


Figure 7 : Generic block diagram for proposed methodology called THE REUSABILITY DESIGN APPROACH- RDA

3.3 Example/ Implementation using the Reusability Design Approach – The RDA

3.3.1 Design and model of a Humanoid Robotic Hand [7]

Based on servo-per-joint model of prosthetic hand, the considered model will have three joints in one manipulator representing knuckles of each finger and four of such manipulators, similar in dimensions to that of human fingers, attached to the fixed base which is morphologically the palm of hand. Human fingers mainly exhibit one dimensional motion which is when they curl towards the palm. Although human fingers also shows limited sideways motion moving to-wards and away from each other like scissors blades but this motion is avoided for simplification. Also attached to this base is thumb exhibiting two dimensional motions, one towards the palm perpendicularly and one towards the fingers parallel to palm. The model consists of an environment base, palm, which is connected to ground on one end and to all the fingers and thumb on other hand. Each finger consists of three bodies and three revolute joints. Two of these joints connect the three bodies representing knuckles of finger and third joint connects the finger to palm.

Same procedure is followed for thumb but thumb has varying joint revolute axis so joint angles will be different. The joint between first knuckle of thumb and palm enables it to rotate parallel to palm whereas remaining two joints makes the other two knuckles revolve perpendicular to palm giving both the motions as seen in the figure below.

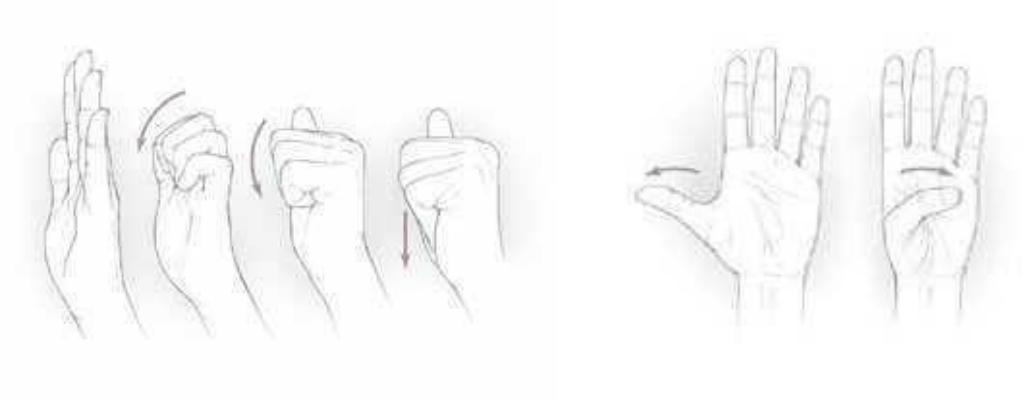


Figure 8 : Human hand motion

This robot manipulator is designed to have four fingers or grippers, and thumb along with Dc servo motors to actuate the movements of the hand. This is because the angular movement of every joint has to be planned to complete the task. There are two joints, three links in every finger and the degree of freedom is four. Once complete data is fed into the controller of the robot, it can execute the task easily. While designing the controller, a lot of other considerations are taken to account like robustness, minimized error, timing etc. Here in this case observer based state feedback controller recommended.

Let us now specify the components and specifications of this robotic hand [7] according to the suggested approach.

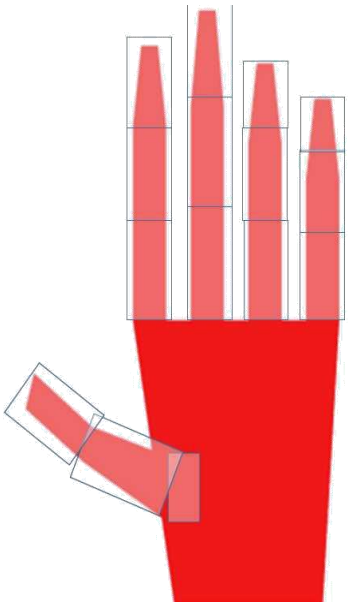


Figure 9 : Robotic hand

Parts of a robotic arm are:

1. Links and joints
2. The controller
3. Actuators
4. Power module

Chopping down the manipulator into its possible appropriate components, we would get the following:

1. Architectural component
2. Control component
3. Power module component

3.3.2 Specifications of components of a Robot Hand manipulator

As mentioned above, the robot manipulator here can be broken into these three components which are architecture, control and power module. Let us look at the architectural component specifications.

3.3.2.1 The Architectural component

The architectural component as the name suggests has the architecture of the system designed. The specifications of the component are broken into formal and informal specifications. We will be discussing the component based system engineering I.e informal specifications first.

3.3.1.1.1 Informal Specifications (Non-Technical Specs.)

In the informal specification, it is to be mentioned how old the architecture is, so that the engineer has an idea of how old the technology is. Also, it should be well explained that what the purpose of the component is, say here, this architectural component is applicable in any of the robotic arm application say for example pick and place robot application industry. The variations this architecture may bare are to be specified as well. The level of reuse is high because the same

architecture maybe reused with very little changes into another similar application. The very little changes need to be mentioned in the design documentation.

- **Context:** *Portable scenario .*
- **Level of Reuse:** *The Component can be plugged into the system at the early design stage.*
- **Agility:** *Defines the age of the component. More it is used in applications, more it is stable and reliable.*
- **Intent of reuse:** *Objective is to design a generic architecture of humanoid robotic hand.*
- **Possible errors (safer reuse):** *This component includes the variations the control may bare, how flexible the control can be.*

Design Considerations and details of the hand – Context (Informal Specs.)

Assumptions:

- It is assumed the robotic hand manipulator is flexible and portable.
- Material considered for this is steel to make it light weight and durable.
- All servos connected to it are assumed to be ideal having no losses.
- Model is assumed to be having ideal friction less joints.
- Inertial tensor matrix is considered diagonal and same for all three axes.
- The design architecture is of an actual industrial pick and place robot manipulator.

3.3.1.1.2 Formal specifications (Technical Specs.)

Formal specifications are divided into two parts. The *internal* specifications of the architectural component include the architecture along with the interactions between

the system, its features and its components also. The *external* features of the component include the software it is built on, say, MATLAB, etc. The architecture is portable because all it tells is the design detailed of what is required in it. It can be modified as well according to the need. It is interoperable, as this component can interact with the other components easily. Figure 9 and 10 shows the detailed model description of the arm.

1. **Internal Specs.**

- **Language:** *Matlab*
- **Type :** *Architectural component*
- **Design description:** *Design details and dimensions.*
- **Architectural aspect**: *Internal structure of the component. The way internal sub components may be interacting.*
- **Encapsulation:** N/A

2. **External Specs.**

- **Function:** *Function is architecture model of the system. It is the first component to be built out of all.*
- **Technology:** *Assumed compatible.*
- **Interoperability:** *Assumed yes*
- **Portability:** *Assumed yes*
- **Integration Framework:** *Specify on what frameworks it can be integrated.*

3.3.1.1.3 Architectural component - Design Description (Formal specs.)

Body:

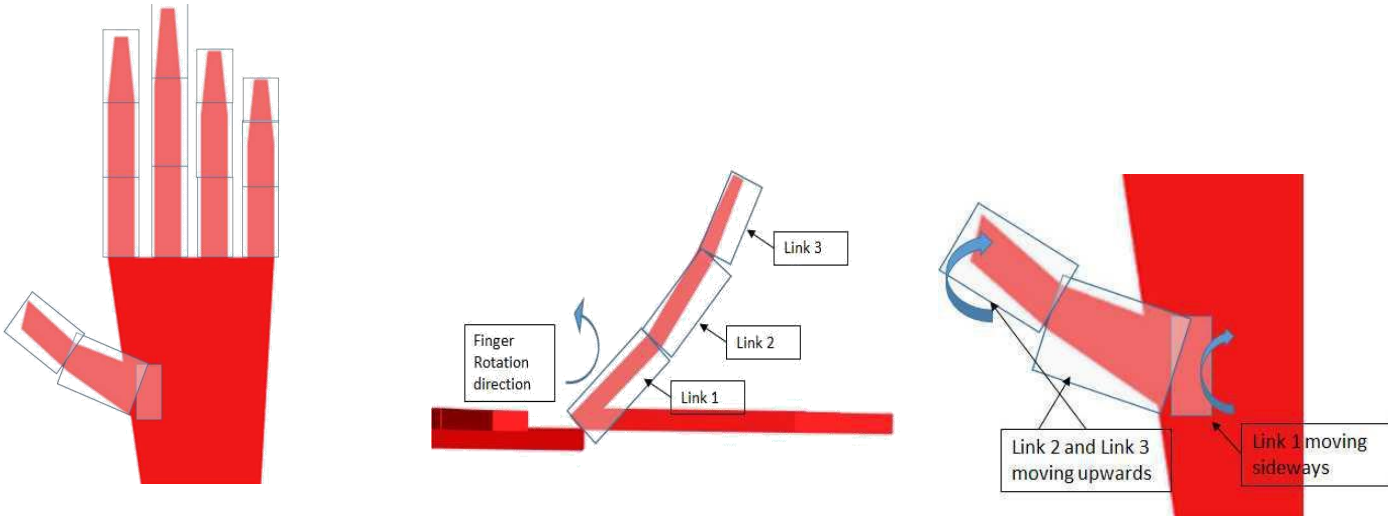
In body toolbar of MATLAB, physical body is made by giving its end co-ordinates or corner axis in terms of Cartesian co-ordinate systems. Other than co-ordinates the block takes mass of the body and its inertia. Center of gravity and joint axis is also mentioned so as to make the motion of the link. In the designing phase links are considered to be of Aluminum due to its light weight having density $2{:}70_{cm}^{\frac{gm}{3}}$. As model is of human hand its dimensions of manipulator links are taken from human hand knuckles itself. Table [2] shows the length, mass and inertia of links of five fingers.

Joints:

Joints toolbox in MATLAB is of two types revolute and prismatic. Revolute joints takes the input axis in terms of Cartesian co-ordinates around which the link has to rotate. Within the joint option there are several utilities like sensors and actuators so as to actuate the joint with torque and senses its angle, velocity, acceleration and torque for feedback. Actuator moves the joint to the provided angle. Here everything is assumed ideal. Physically joint moves with motors having damping and other losses but for ease conditions assumed are ideal.

Link	Label	Mass (gm)	Inertia $(gm.cm^2)$	Length (cm)
Fore finger	11	12.15	11.39	3
	12	10.125	7.178	2.5
	13	11.1375	9.106	2.25
Middle finger	11	12.15	11.39	3
	12	10.125	7.178	2.5
	13	11.1375	9.106	2.25
Ring Finger	11	12.15	11.39	3
	12	10.125	7.178	2.5
	13	11.1375	9.106	2.25
Small finger	11	12.15	11.39	3
	12	10.125	7.178	2.5
	13	11.1375	9.106	2.25
Thumb	51	1.35	0.478	2
	52	10.125	12.867	2.5
	53	6.755	5.765	2

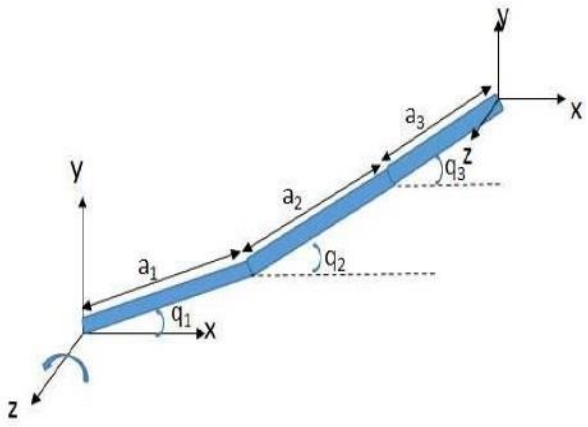
Table 3: Architectural component - Design Description (Formal specs.)



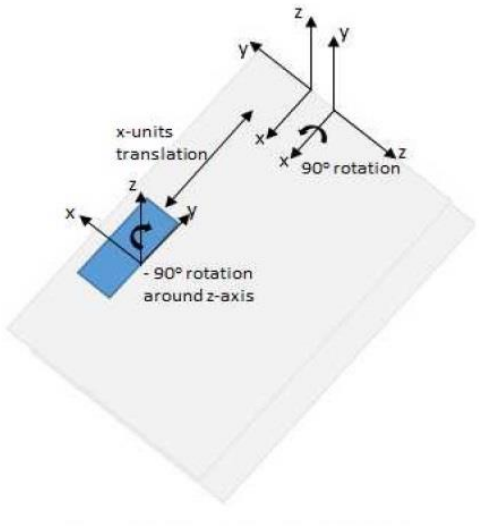
A. B. C.

A. Design of the system. Modeled hand each box representing one link and at inter connection of link is a joint
B. Side View of Model showing motion of finger in one direction only
C. Front view showing thumb motion where one link connected to palm rotates parallel to palm where as other two revolve perpendicular to palm

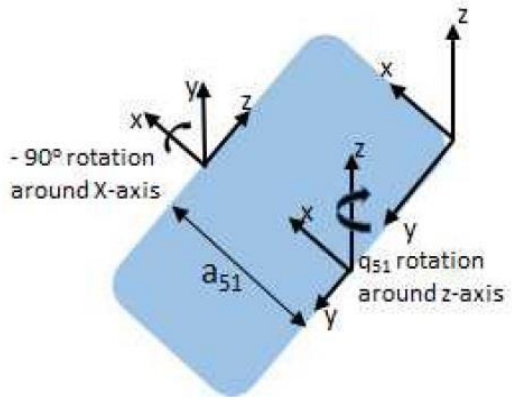
Figure 10: Architectural Component- Model of the arm- Design Description (Formal/Internal specs)



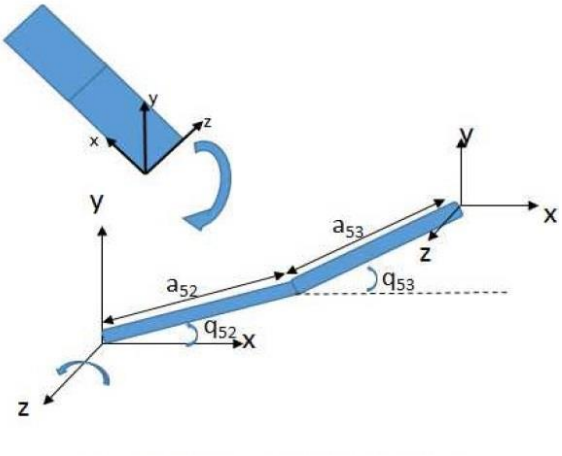
A. Unit finger rotation



B. Thumb base link rotation



C. Thumb first link rotation



D. Thumb second and third link rotation

Figure 11 : Architectural Component – Architectural Aspect (Formal/Internal specs)

Design Description - Model calculations: Formal Specs.

Unit finger:

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1^0 = A_1 = \begin{bmatrix} cosq_{11} & -sinq_{11} & 0 & a_{11}cosq_{11} \\ sinq_{11} & cosq_{11} & 0 & a_{11}sinq_{11} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^1 = A_2 = \begin{bmatrix} cosq_{12} & -sinq_{12} & 0 & a_{12}cosq_{12} \\ sinq_{12} & cosq_{12} & 0 & a_{12}sinq_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = A_3 = \begin{bmatrix} cosq_{13} & -sinq_{13} & 0 & a_{13}cosq_{13} \\ sinq_{13} & cosq_{13} & 0 & a_{13}sinq_{13} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Net Transformation matrices for all four fingers:

$$T_3^0 = A_3 * A_2 * A_1 = \begin{bmatrix} c_{123} & -s_{123} & 0 & a_{11}c_1 + a_{12}c_{12} + a_{13}c_{123} \\ s_{123} & c_{123} & 0 & a_{11}s_1 + a_{12}s_{12} + a_{13}s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ First finger}$$

$$T_3^0 = \begin{bmatrix} c_{123} & -s_{123} & 0 & a_{21}c_1 + a_{22}c_{12} + a_{23}c_{123} \\ s_{123} & c_{123} & 0 & a_{21}s_1 + a_{22}s_{12} + a_{23}s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ Second finger}$$

$$T_3^0 = \begin{bmatrix} c_{123} & -s_{123} & 0 & a_{31}c_1 + a_{32}c_{12} + a_{33}c_{123} \\ s_{123} & c_{123} & 0 & a_{31}s_1 + a_{32}s_{12} + a_{33}s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ Third finger}$$

$$T_3^0 = \begin{bmatrix} c_{123} & -s_{123} & 0 & a_{41}c_1 + a_{42}c_{12} + a_{43}c_{123} \\ s_{123} & c_{123} & 0 & a_{41}s_1 + a_{42}s_{12} + a_{43}s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ Fourth finger}$$

Thumb:

$$A_0 = \begin{bmatrix} 0 & 1 & 0 & x \\ 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ Base link}$$

$$A_1 = \begin{bmatrix} cosq_{51} & 0 & -sinq_{51} & a_{51}cosq_{51} \\ sinq_{51} & 0 & cosq_{51} & a_{51}sinq_{51} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$, First link

$$A_2 = \begin{bmatrix} cosq_{52} & -sinq_{52} & 0 & a_{52}cosq_{52} \\ sinq_{52} & cosq_{52} & 0 & a_{52}sinq_{52} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} cosq_{53} & -sinq_{53} & 0 & a_{53}cosq_{53} \\ sinq_{53} & cosq_{53} & 0 & a_{53}sinq_{53} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$,Second and Third link

The Net Transformation matrix of thumb:

$$T_3 = A_3 * A_2 * A_3 * A_0$$

$$T_3 = \begin{bmatrix} s_1c_{23} & -s_1s_{23} & c_1 & x + a_{53}s_1c_{23} + a_{52}s_1c_2 + a_{51}s_1 \\ s_{23} & c_{23} & 0 & a_{53}s_{23} + a_{52}s_2 \\ -c_1c_{23} & c_1s_{23} & s_1 & a_{51}c_1 + a_{52}c_1c_2 + a_{53}c_1c_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Velocity Jacobians:

$$J_{vc1} = \left[\frac{dvc1}{dq1} \quad \frac{dvc1}{dq2} \quad \frac{dvc1}{dq3}\right]$$

$$J_{vc1}^{T} = \begin{bmatrix} \frac{1}{2}a_{11}s_1 & \frac{1}{2}a_{11}c_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$J_{vc1} = \left[\frac{dvc2}{dq1} \quad \frac{dvc2}{dq2} \quad \frac{dvc2}{dq3}\right]$$

$$J_{vc2}^{T} = \begin{bmatrix} -a_{11}s_1\frac{1}{2}a_{12}s_{12} & a_{11}s_1 + \frac{1}{2}a_{12}s_{12} & 0 \\ -\frac{1}{2}a_{12}s_{12} & \frac{1}{2}a_{12}c_{12} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This explains technical aspect of the component.

3.3.1.2 The Control component:

3.3.1.2.1 Informal Specifications (Non-Technical Specs.)

Informal specifications tell when the component was built, and the purpose of the component. The purpose of this component is to build and design the control code for a humanoid hand robot manipulator which is applicable to pick and place applications as well. This component includes the variations the control may bare, how flexible the control can be.

This component may be plugged into any other similar applications which make the level of reuse high.

Assumptions:

- The component built communicates with other components.
- The controller designed is robust and tested.
- The manipulator communicates with the entire system when connected.

For this MIMO system containing three coupled inputs and three outputs gain matrices are formed using pole placement command in MATLAB and poles are placed at -2+2j , -2-2j , -7+25j , -7-25j , -5+18j and -5-18j. So, for these values of poles gain matrix is obtained for 6 states of the system. The controls tested are state feedback, error integrator and observers based state feedback. The error integrator, results obtained contains very large overshoots as does PID (tuned) in simulation. So, state feedback controller is applied.

- **Context**: *Portable scenario.*
- **Level of Reuse:** *Once the architecture of a system is made. Control component can be plugged into the system.*
- **Agility:** *Age of component built in 2016.*
- **Intent of reuse:** *Objective is to build up control for hand with 2 control movements.*
- **Possible errors (safer reuse):** *This component includes the variations the control may bare, how flexible the control can be.*

3.3.1.2.2 Formal Specifications: (Technical Specs.)

This component is the main controller of the robotic arm. The formal specifications are divided into internal and external specifications. The internal specifications may be detailed as follows.

It consists of all the matrices which are required to design the control for the given conditions as per a system requirement. This component includes the control of the arm; say observer based state feedback, which is chosen as the robust controller is a part of this component. This component falls into the type of “code component” because it contains the control code for robot manipulator. In this scenario, this code component is linked with the architectural component and power module.

The external specifications of the control component include details like the software it is built on, for example here on MATLAB. This component is compatible with MATLAB software and this can be saved in libraries to drop down as a reusable entity. It is interoperable, as this component can interact with the other components easily as assumed.

1. **Internal specs.**

I. **Language:** *Matlab*

II. **Type** : *Code component*

III. Design description:

IV. *Joint matrices,*

V. *Transformation matrices,*

VI. *Velocity Jacobians*

VII. *The control of the arm(say integral control with state feedback, which is chosen as the robust controller)*

VIII. **Architectural aspect:** *internal structure of the component. The way internal sub components may be interacting.*

IX. **Encapsulation:** *N/A*

2. External specs.

I. **Function:** *Function is design of control. This component effects the complete system as this is the main control of the robotic hand.*

II. **Technology:** *Assumed compatible.*

III. **Interoperability:** *Assumed yes*

IV. **Portability:** *Assumed yes*

V. **Integration Framework:** *Specify on what frameworks it can be integrated.*

Design Description (Formal/Internal Specs.)

Note: For simplification a small part of the control calculation is shown for understanding.

Control of the system: State feedback controller

$u = \mathrm{M(q)}[-K1\dot{q} - k2\dot{q}] + C(q,\dot{q})\dot{q} + \mathrm{G(q)}$ ..... (1)

Linearized inertial and Gravitation matrix:

$$M = \begin{bmatrix} 725 & 329 & 91 \\ 329 & 177 & 54 \\ 91 & 54 & 23 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} 0.01 & -0.023 & 0.014 \\ -0.023 & 0.072 & -0.079 \\ 0.014 & -0.079 & 0.173 \end{bmatrix}$$

$$G = \begin{bmatrix} 1350 & 530 & 125 \\ 530 & 530 & 125 \\ 125 & 125 & 125 \end{bmatrix}$$

In state space form, $A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 532630 & 201320 & 60650 & 0 & 0 & 0 \\ 218680 & 78060 & 31580 & 0 & 0 & 0 \\ 46200 & 17970 & 8640 & 0 & 0 & 0 \end{bmatrix}$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.01 & -0.023 & 0.014 \\ -0.023 & 0.072 & -0.079 \\ 0.014 & -0.079 & 0.173 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

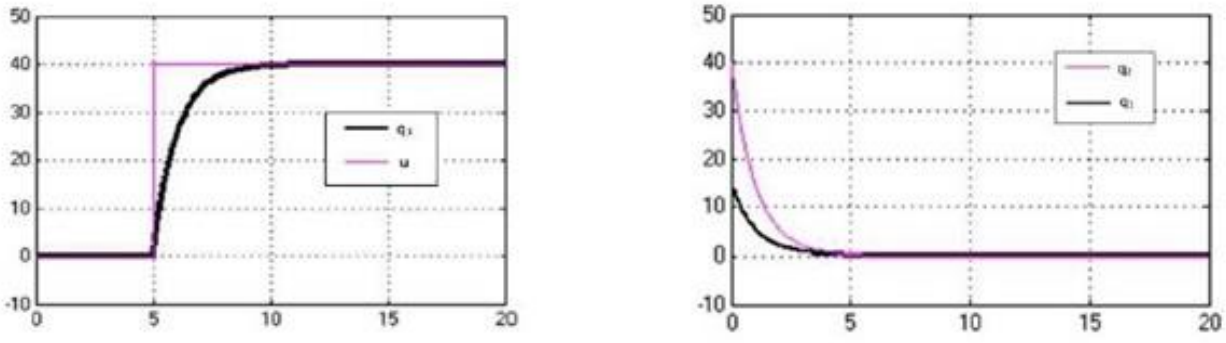


Figure 12: Control Component - Design Description (Formal/Internal Specs.)

3.3.2.3 The Power module component

This component has not been designed for the example above as it is not built physically.

When physical implementation is done, we need to specify the power module component's voltages and power at which the system works. This component provides the energy to the drive controller and actuator as well. There are three types of power supplies, electrical, hydraulic and pneumatic. For an industrial robot manipulator we have electrical power supply. This is the internal specification.

3.3.3 Reusability Analysis:

When the systems are developed in a component based environment, hardware and software both can be managed to provide maximum reusability. The Reusability Design Approach (RDA) lays emphasis on developing systems in a way that we can reuse our systems if and when required in future. One of the biggest advantages of a component based system is that it makes the system components reusable such that they can be built as an independently executable entity, an independent block which makes them best suitable for reuse into other applications. This approach, in the long run, also makes the tedious task of assembling and integrating components easier as it reduces the number of parts needed for construction of a system. There definitely is a good chance that many of the components developed for one system might not be required in future for other applications, in that case components will die its own death. It obviously depends on how large or small the system application is but the essence doesn't change. It is not confined to a particular application. Larger the system, larger would be its requirements.

The robot manipulator here, which is just a small representation of the proposed idea, is built keeping in mind the RDA, and has been chopped down into constituent components which can be used into other systems if they fit into the requirement. Other than that, the manipulator can also be treated as one component as a whole, with its defined sub components and can be applicable in another application. This may require a few modification and improvement in the system as the new system demands. But with minor adjustments, this can be made possible. For example, if there is an application where there is need of a humanoid hand writing assistant, a pick and place robot used in games, a pick and place robotic hand in medical applications, or even as a kitchen helper. The same manipulator can be used with addition of components into it. All depends on the requirement of that particular system.

The proposed methodology is not confined only to software or just hardware of the industrial control systems. Our approach is combination of component based software and/or hardware engineering subsystems to achieve the desired reusability tasks.

3.3.4 Example 2:

The robot manipulator shown below is a valid example to show how the same pick and place manipulator can be applicable in three different applications with slight modifications.

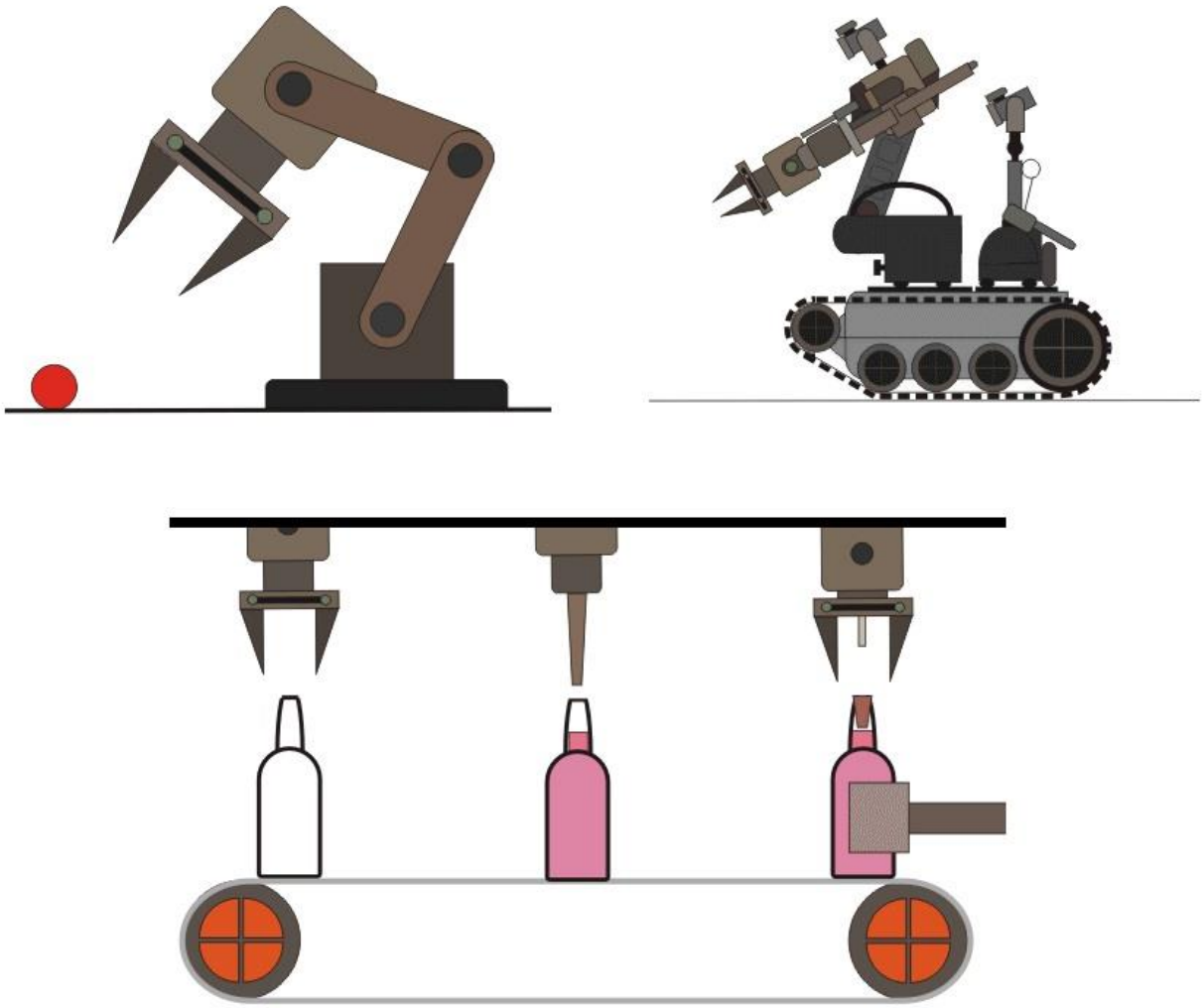


Figure 13: Example of another Robot manipulator applicable in 3 different applications.

CHAPTER 4

MEASURING REUSABILITY

The idea of reusing software components has been present in software engineering for several decades; component reuse is still facing numerous challenges. Reusability is an attribute which is impossible to be measured through any straight forward method. There are a lot of factors which affect the quality of systems hence affect the measure of reusability. Our reusability measurement model presented in this thesis does not aim on identifying just new characteristics of components determining reusability while rejecting the existing ones, but rather focuses on structuring them better and identifying a common superset of these characteristics to determine the reusability of components of our system.

Therefore, for our component based system we define a few factors that tell the relationship between components using predefined index numbers to measure the reusability of our system. This is implemented on MATLAB using the fuzzy logic approach. This chapter describes factors which affect the reusability of systems and how can we measure reusability using fuzzy logic.

4.1 Proposed Model to Measure Reusability

Considering the systems and components ready, we are now explaining a few factors out of many, which will influence the measure of reusability. Following metrics are to be considered,

1. Customizability
2. Interface complexity
3. Understandability and Documentation
4. Maintainability (Portability)
5. Commonality

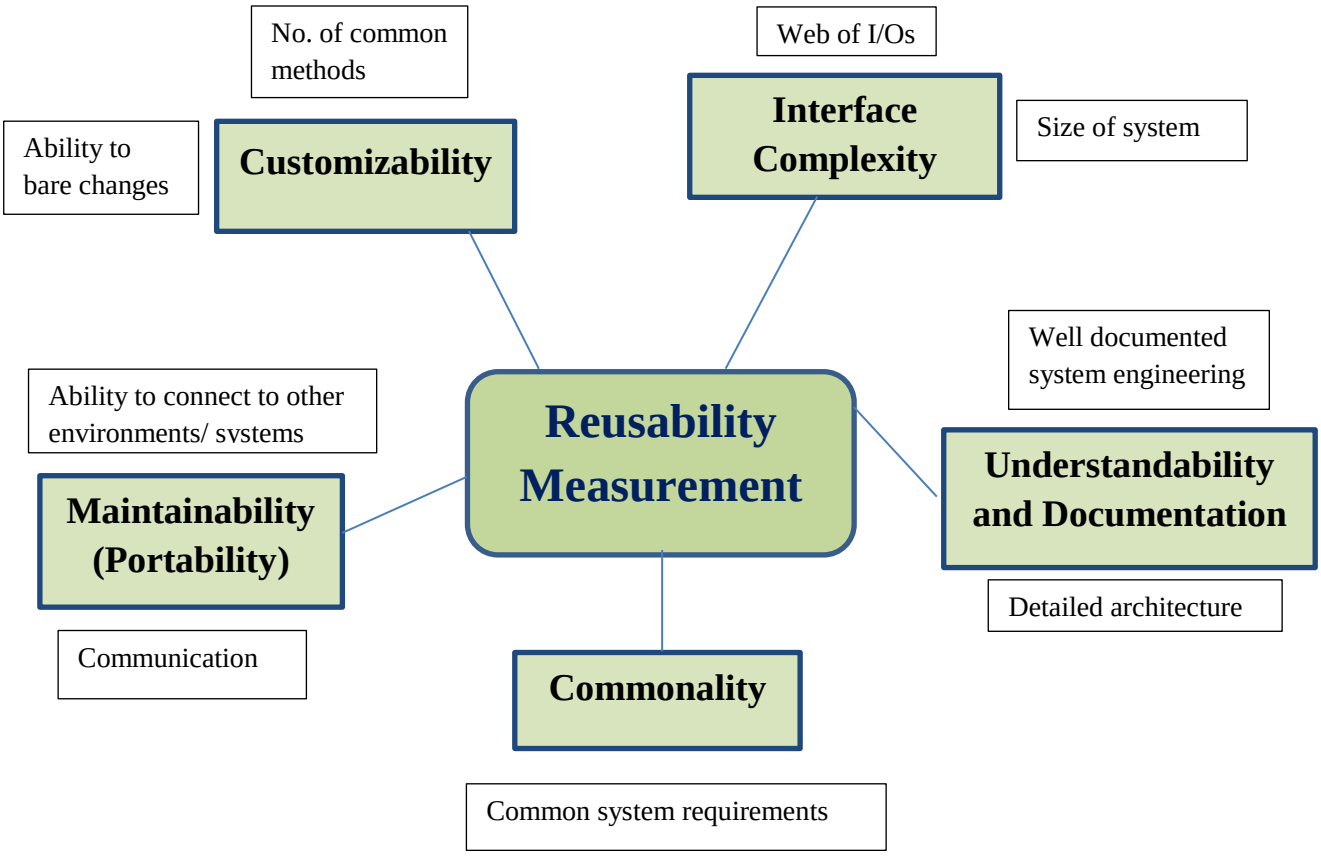


Figure 14 : Reusability Measurement

4.1.1 Customizability

Customizability is the defined to be the measure of ability to modify the component as per the requirements of the system. This means, more you are able to customize the component, higher is the reusability. The formula for customizability is given as [5],

Customizability = (No. of set Methods) / (Total no. of Properties)

For our understanding we have chosen it in MATLAB to be in the range which may vary from 0 to 1.

4.1.2 Interface Complexity

Interface complexity is the measure of how complex the system is. But how do we find that out? There are methods to show the complex architecture of the systems. For example, graphs. Connected graphs show the behavior and interaction of the components that tell the interface complexity in a system. The most common complexity metric introduced by McCabe(1976), is based on program graph and is called the Cyclomatic Complexity, is defined as:

$$V(G) = e - n + 2p$$

Where,
e is the number of edges,
n is the number of nodes in the graph, and
p is the number of connected components.

For ease, we have chosen it in MATLAB to be in the range which may vary from 0 to 1.

4.1.3 Understandability and Documentation

Understandability metric is another important metric to measure reusability. Whether the source code is reachable or not, proper documentation of the system components makes it easy for the user. The term documentation here refers to a couple of factors that are, component manuals, marketing information, details of architecture etc. a very good quality document also includes the functional and system description. It helps understanding the component and it helps maintaining activities in an effective manner. For the present work, we break understandability into low, medium and high categories.

4.1.4 Commonality

Commonality metric defines how well the functionality of the components matches with the functionality of the system. If a component provides very superior functionalities but very less commonality, the reusability of the component will be very low. For the present work, we break understandability into low, medium and high categories.

4.1.5 Maintainability (Portability)

Portability is defined as the ability to be transferred from one environment into another with or without modifications. If a component is portable, it means that it can be shifted from one framework to another without encountering unsolvable errors. For better reusability, like all the above factors, portability also plays an important role for the expansion of implementation of components, which means it should have the ability to fit into other operating systems as well. If that's the case, reusability is high. For the present work, we break understandability into low, medium and high categories.

4.2 Sheza Reusability Index (SRI):

We have defined Reusability Index, named ***Sheza Reusability Index SRI*** for the resulting Reusability measurement ranging from 0-1. We have five conditions as a result of reusability measurement which are;

1. Very Low
2. Low
3. Medium
4. High
5. Very High

Given as,

Reusability	**Sheza Reusability Index - SRI**
Very Low	0-0.2
Low	0.21-0.39
Medium	0.4-0.6
High	0.61-0.8
Very High	0.81-1

Table 4 : Sheza Reusability Index

For this, we have 3^5 i.e 243 possible combination of inputs considered to design the rule base.

In our work, we have shown 12 combinations for ease.

The rules are;

1. If customizability of components is low, interface complexity is high, understandability is low, commonality is low and maintainability is low then Reusability will be very low.

2. If customizability of components is low, interface complexity is high, understandability is low, commonality is medium and maintainability is medium then Reusability will be low.

3. If customizability of components is low, interface complexity is high, understandability is low, commonality is low and maintainability is high then Reusability will be very low.

4. If customizability of components is medium, interface complexity is low, understandability is medium, commonality is high and maintainability is medium then Reusability will be medium.

5. If customizability of components is medium, interface complexity is low, understandability is medium, commonality is high and maintainability is high then Reusability will be low.

6. If customizability of components is medium, interface complexity is medium, understandability is low, commonality is medium and maintainability is high then Reusability will be low.

7. If customizability of components is medium, interface complexity is low, understandability is high, commonality is high and maintainability is medium then Reusability will be medium.

8. If customizability of components is high, interface complexity is medium, understandability is medium, commonality is high and maintainability is high then Reusability will be high.

9. If customizability of components is high, interface complexity is low, understandability is high, commonality is high and maintainability is high then Reusability will be very high.

10. If customizability of components is high, interface complexity is undefined, understandability is high, commonality is high and maintainability is high then Reusability will be high.

11. If customizability of components is medium, interface complexity is medium, understandability is low, commonality is low and maintainability is low then Reusability will be low.

12. If customizability of components is low, interface complexity is low, understandability is low, commonality is low and maintainability is high then Reusability will be low.

S.no	Customizability	Interface complexity	Understandability	Commonality	Maintainability	Reusability
1	Low	High	Low	Low	Low	Very Low
2	Low	High	Low	Medium	Medium	Low
3	Low	High	Low	Low	High	Very Low
4	Medium	Low	Medium	High	Medium	Medium
5	Medium	Low	Medium	High	High	Low
6	Medium	Medium	Low	Medium	High	Low
7	Medium	Low	High	High	Medium	Medium
8	High	Medium	Medium	High	High	High
9	High	Low	High	High	High	Very High
10	High	Undefined	High	High	High	High
11	Medium	Medium	Low	Low	Low	Low
12	Low	Low	Low	Low	High	Very Low

Table 5 : Rules in tabular form

4.3 The Fuzzy Model in MATLAB

As mentioned above, this work emphasizes on the five factors to measure reusability of a system. So, for this fuzzy model we have five inputs namely, customizability, interface complexity, understandability, commonality and maintainability. Each input has three membership functions defining ranges from 0 to 1 for low, medium and high categories. The rule base provides the crisp values of reusability. The output which is reusability has five membership functions for very low, low, medium, high and very high categories.

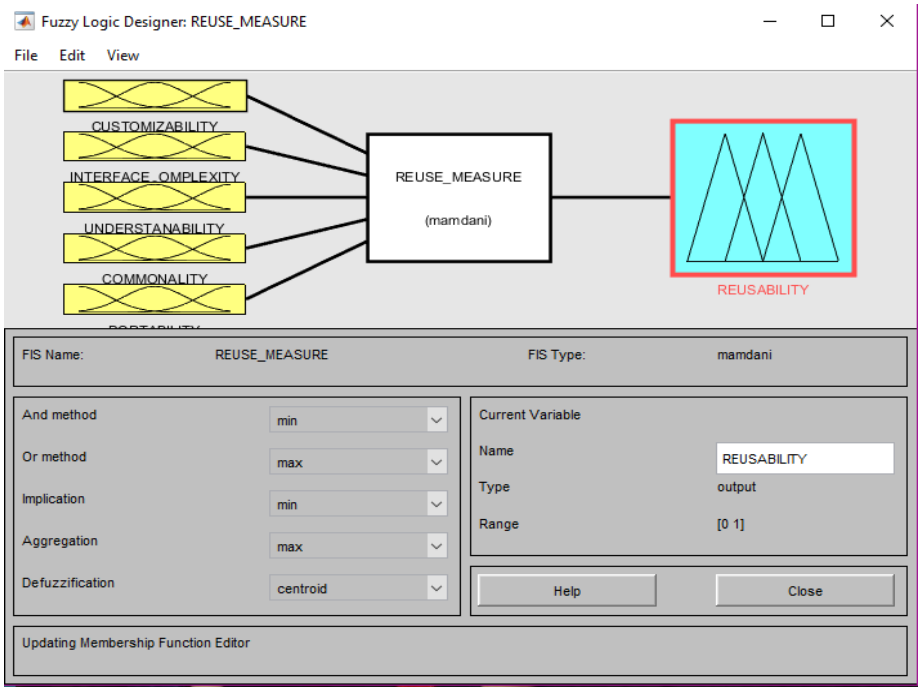


Figure 15: Fuzzification Model

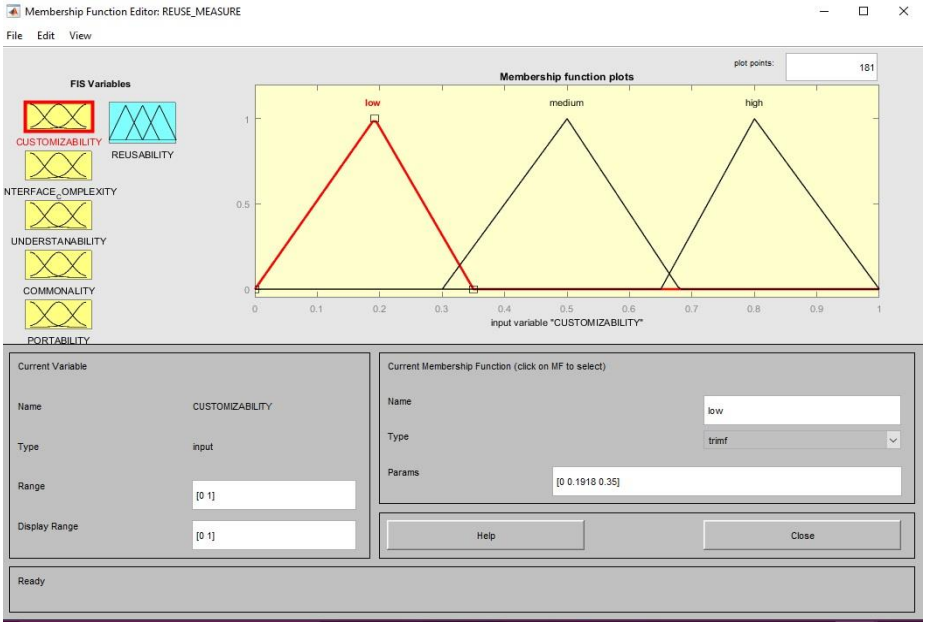


Figure 16: First Input with its three Membership Functions

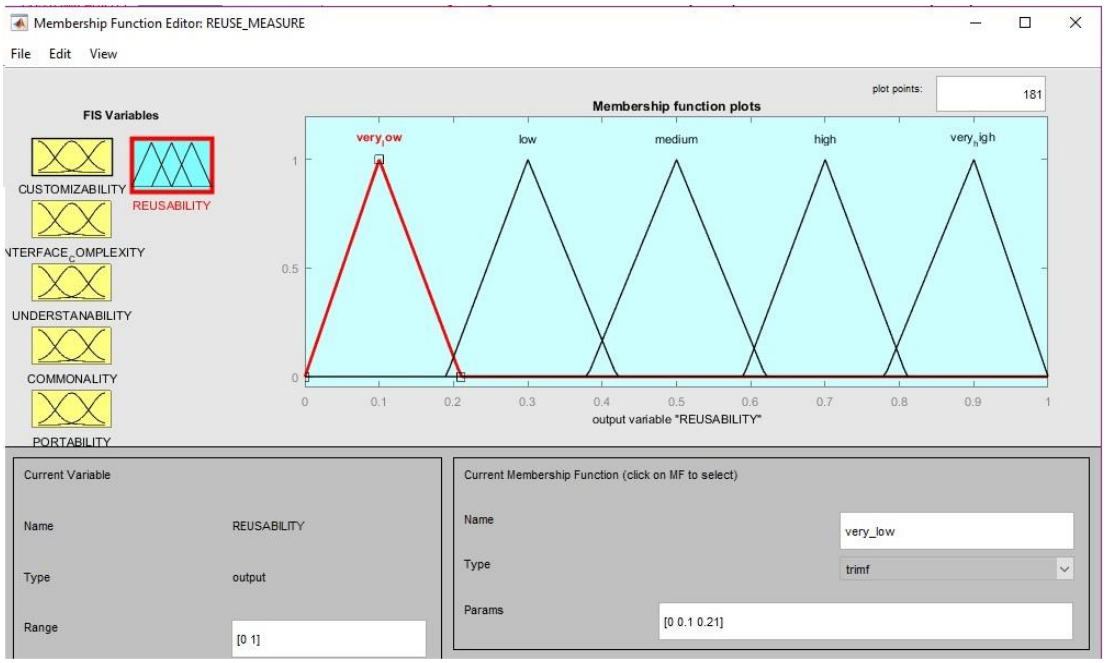


Figure 17 : Output with its five membership functions

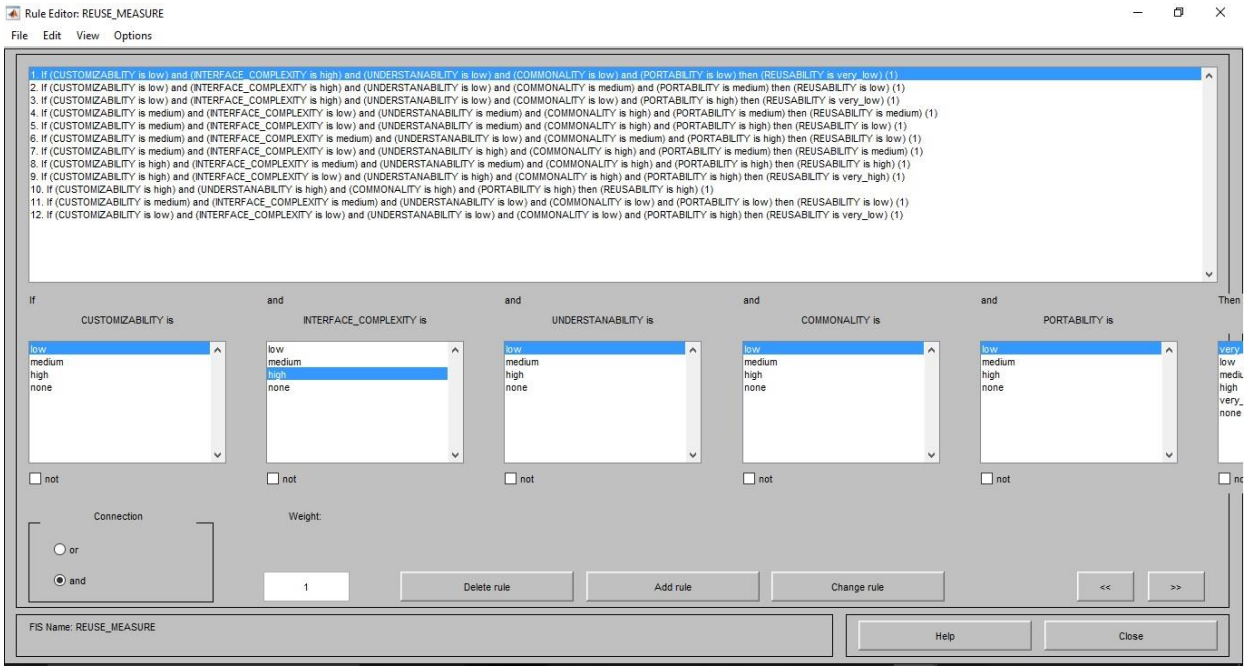


Figure 18 : Rules

4.4 Results

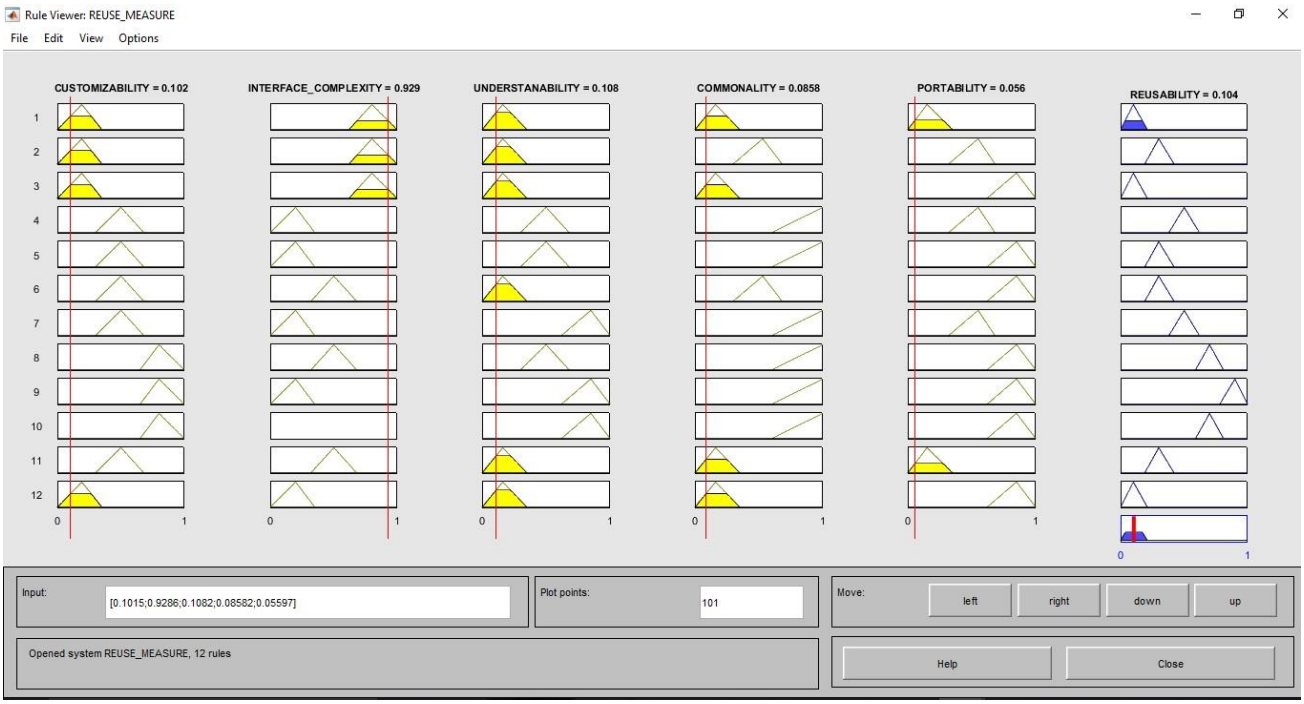


Figure 19 : Output Sheza Reusability Index (SRI) =0.18 (REUSABILITY is Very Low)

Results show that when customizability, Maintainability, commonality, understandability DECREASES and interface complexity INCREASES, reusability is VERY LOW.

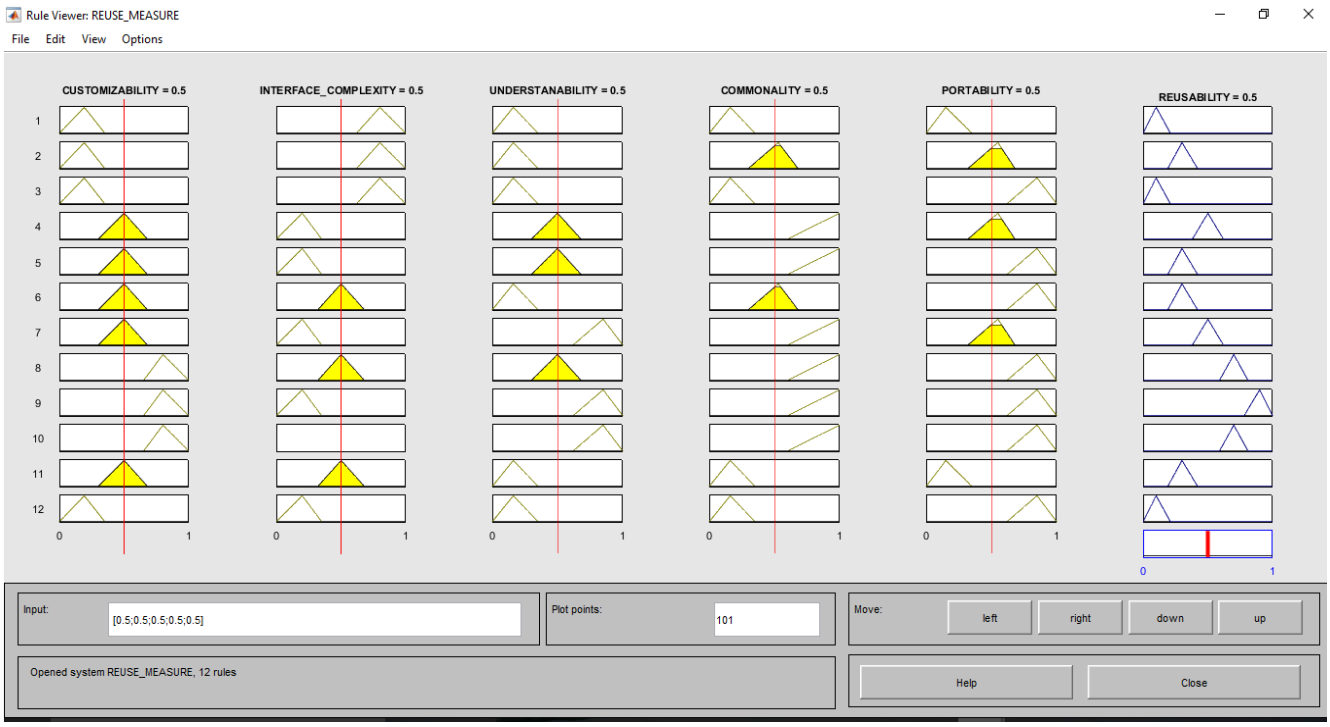


Figure 20 : Ruler View of Results showing output i.e. Sheza Reusability Index (SRI) = 0.5

Results show that when customizability, Maintainability, commonality, understandability and interface complexity fall in an average scale, Reusability is MEDIUM

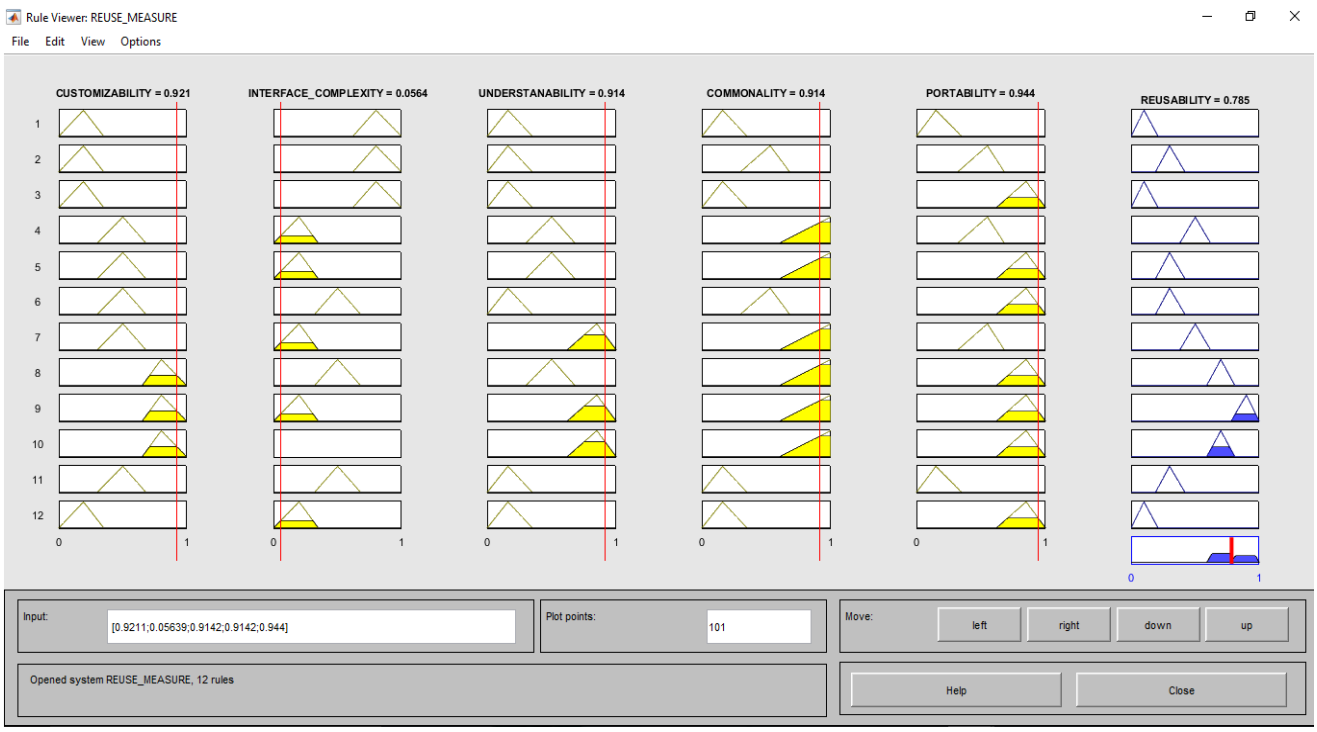


Figure 21 : Ruler View of Results showing output i.e. Sheza Reusability Index (SRI) =0.785

Results show that when customizability, understandability, commonality, and maintainability increases, and interface complexity decreases, Reusability is HIGH.

CHAPTER 5

CONCLUSION

Component based engineering (CBE) is a branch of engineering which is divided into two categories namely, Component based software engineering (CBSE) and Component based systems engineering (CBSysE). It deals with development and maintenance of control systems by modern technology of componentization. Componentization of control system means designing a control system by chopping down the system into functional blocks so that it gets easier to understand and operate the system in the long run. As far as the CBSysE is concerned, reusability can be implemented on the design level. This is the basic point to focus for component based system engineering. It means that when the system engineers are designing a particular system, knowing all the design specifications, all the assumptions, the complete model which should include properly and thoroughly documented design rationale. They can actually work on finding the reusable parts of that system, be it hardware of software. “The way the components are architected has allowed us a lot of flexibility and customization”, Ferenc Molnár, Autolog, Inc. CBSE has been implemented a couple of systems by building up component frameworks. But when a combination of CBSE and CBSysE is implemented, it can yield into a more robust, fast, safe, reliable system, making reusability more useful, as suggested in our work.

We have described in detail the generic framework and characteristics to build up reusable components by the proposed design technique called the Reusability Design Approach, RDA, followed by an example which has its own limitations and assumptions described at the end of chapter 3. Specifically, we formulate a procedure through which we evaluate the system for reusability of its components that may be used for design of a new system.

In chapter 4, we have described how to measure Reusability using fuzzy logic system in MATLAB, which will help in estimating the quality for the application. This is done by proposing some factors which may affect the measure of reusability of a system followed by defining Sheza Reusability Index as a unit of reusability.

The work proposed here can be used by researchers for further study and empirical validation of the suggested approach along with these existing metrics to measure reusability.

FUTURE RECOMMENDATION

Reusability is likely to have a bright future and a remarkable work for research.

The proposed concept has been shown on a Humanoid Robotic hand as an example. In future this can be implemented on large scale systems along with the architecture and design implementation using the proposed methodology. It would be an achievement if this framework is further considered to be tested on larger industrial system designs to save time and effort of the engineers. Also, In future, the Proposed Reusability Measurement model can also be tested on real systems.

The Reusability Design Approach RDA framework can be a built for a specific Industrial design system or the RDA can be added into Matlab as a framework as well. For example software called SPECTRM-RL is designed specifically using systems engineering approach to build any system with generic models, currently applied successfully on space craft design to show reusability.

This definitely would be a tedious task to start, but would be beneficial in the long run.

REFERENCES

[1] H. Mili, F. Mili, A. Mili, "Reusing software: Issues and Research" Directions, IEEE Transacttion on Software Engineering, Vol. 21, Issues 6, pp:528-561.1995

[2] Euler, Edward **A.,** Steven **D.** Jolly, and H.H. 'Lad' Curtis. "The Failures of the Mars Climate Orbiter and Mars Polar Lander: **A** Perspective from the People Involved." *Advances in The Aeronautical Sciences, Guidance and Control.* Volume **107,**2001.

[3] Ravikumar Mourya, Amit Shelke, Sourabh Satpute, Sushant Kakade, Manoj Botre. "*Design and Implementation of Pick and Place Robotic Arm "* International Journal of Recent Research in Civil and Mechanical Engineering (IJRRCME) Vol. 2, Issue 1, pp: (232-240), Month: April 2015 – September 2015 .

[4] Nancy G. Leveson. Intent Specifications: An Approach to Building Human-Centered Specifications. *IEEE Transactions on Software Engineering*, Vol. SE-26, No. 1, January 2000.

[5] A. Sharma: " Design and Analysis of Metrics for Component based software systems" Ph.D. thesis. Tharpar University Patiala, 2009.

[6] Kathryn Anne Weiss, Elwin C. Ong, and Nancy G. Leveson Massachusetts Institute of Technology "Reusable Specification Components for Model-Driven Development*" 2003*

[7] Sidra Jabeen Yousuf, "Modelling and Control of Robotic Hand" .National University of Sciences and Technology, PNEC. June 2016.

[8] Andreas Speck, "Reusable Industrial Control Systems" , IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 50, NO. 3, JUNE 2003

[9] Frank Lüders, Ivica Crnkovic, Andreas Sjögren; "Case Study: Componentization of an Industrial Control System" IEEE Computer Society Press 2009.

[10] Danail Hristov, Oliver Hummel, Mahmudul Huq, Werner Janjic, "Structuring Software Reusability Metrics for Component Based Software Development", Software Engineering Group University of Mannheim , Germany. CSEA 2012 : The Seventh International Conference on Software Engineering Advances

[11] Sarbjeet Singh, Manjit Thapa, Sukhvinder singh and Gurpreet Singh, "Software Engineering - Survey of Reusability Based on Software Component", Department of Computer Science, Sri Sai College of Engg. & Tech. Badhani (Pathankot). International Journal of Computer Applications (0975 – 8887) Volume 8– No.12, October 2010

[12] Alan W. Brown University of Surrey, Kurt C. Wallnau Carnegie Mellon University, "Engineering of components based systems". CONFERENCE PAPER JANUARY 1996.

[13] CIS 376, Bruce R. Maxim , UM-Dearborn "Software Reuse and Component-Based Software Engineering, "Software Reuse and Component-Based Software Engineering" lecture.

[14] Conference on Technological future conducted in Nov 2016 https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Reusable+Components+at+DIGITEC+2016%3A+Digital+Future

ANNEX A

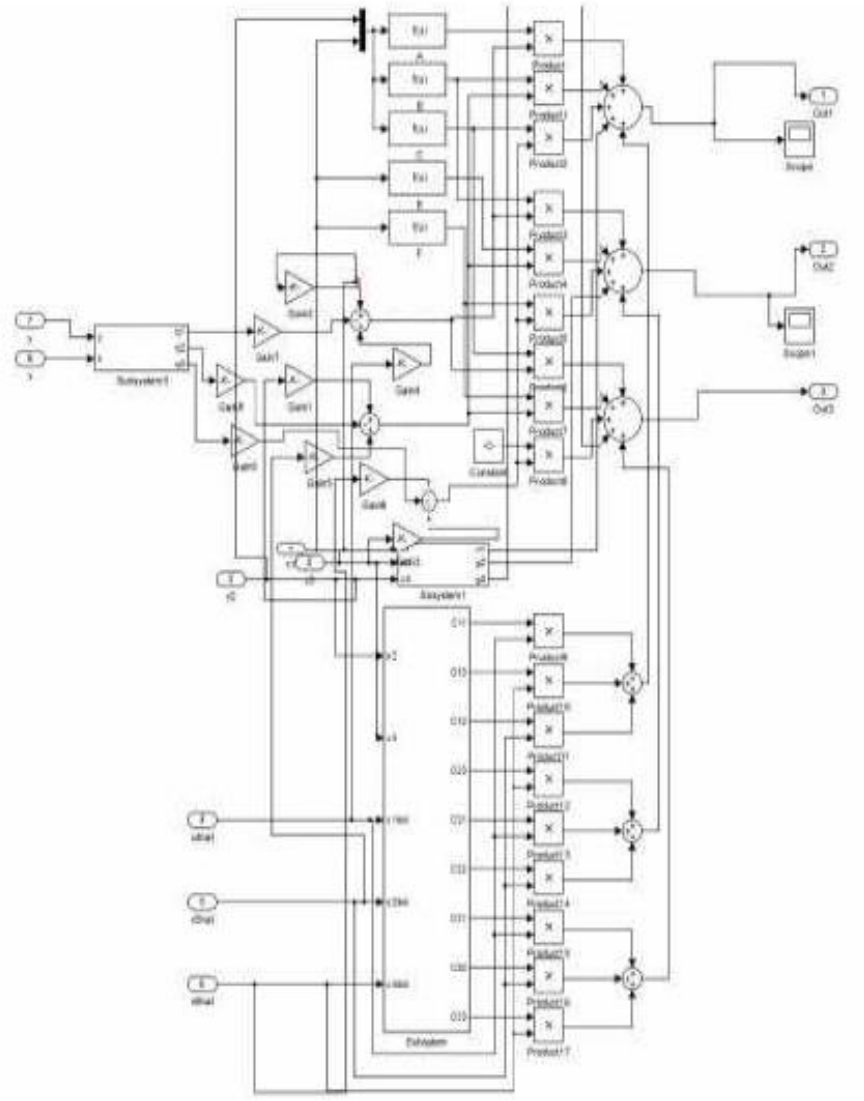


Figure 22 : Control of hand

ANNEX B

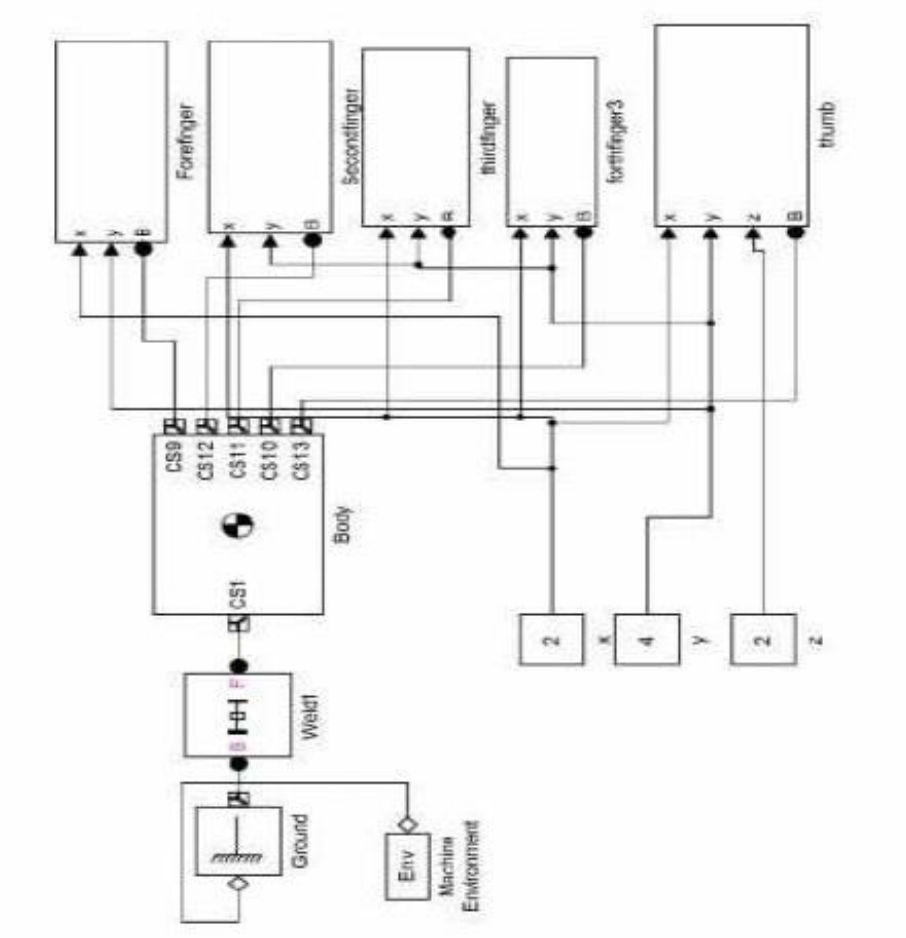


Figure 23 : Main Block of system showing connection of all fingers to palm and environment

ANNEX C

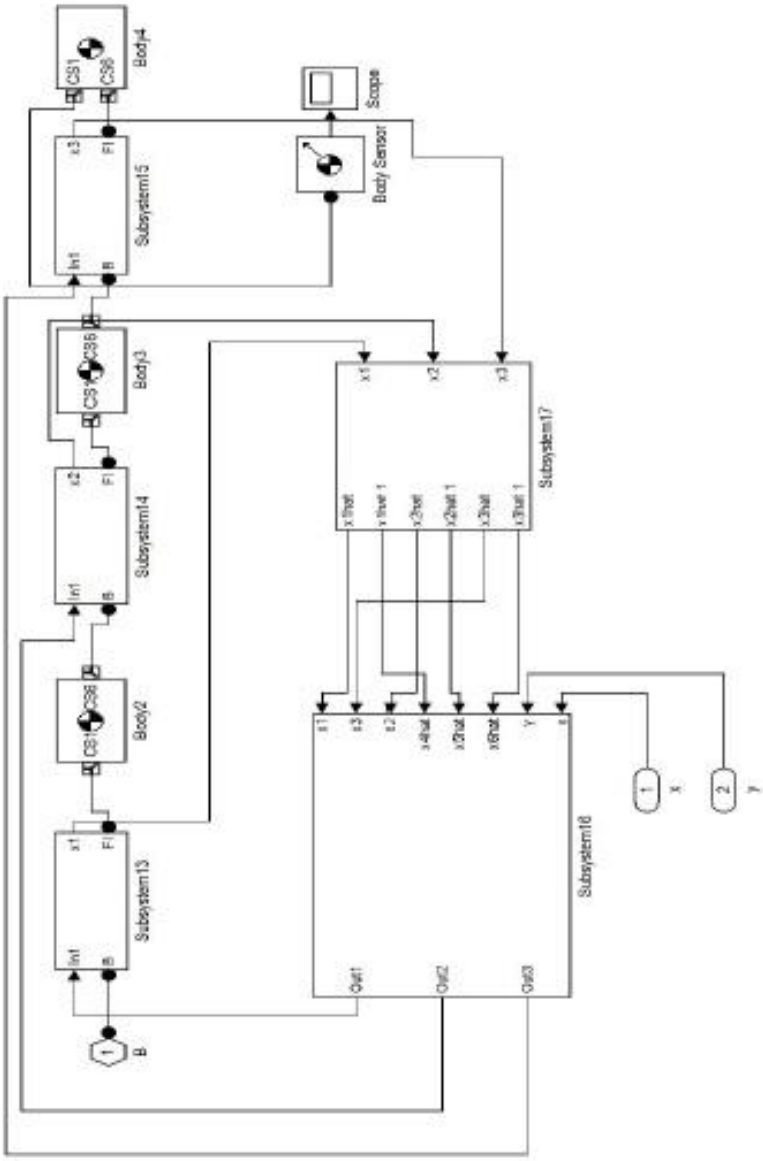


Figure 24 : Subsystem Block of Forefinger (block is same for all fingers, dimensions of body is different)

ANNEX D

Workspace for parameters set for inputs, their membership functions and output fuzzification.

```
[System]
Name='REUSE_MEASURE'
Type='mamdani'
Version=2.0
NumInputs=5
NumOutputs=1
NumRules=12
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='CUSTOMIZABILITY'
Range=[0 1]
NumMFs=3
MF1='low':'trimf',[0 0.191798941798942 0.35]
MF2='medium':'trimf',[0.3 0.5 0.68]
MF3='high':'trimf',[0.65 0.8 1]

[Input2]
Name='INTERFACE_COMPLEXITY'
Range=[0 1]
NumMFs=3
MF1='low':'trimf',[0 0.2 0.35]
MF2='medium':'trimf',[0.32 0.5 0.68]
MF3='high':'trimf',[0.62 0.8 1]

[Input3]
Name='UNDERSTANABILITY'
```

Range=[0 1]
NumMFs=3
MF1='low':'trimf',[0 0.16 0.35]
MF2='medium':'trimf',[0.3 0.5 0.68]
MF3='high':'trimf',[0.62 0.85 1]

[Input4]
Name='COMMONALITY'
Range=[0 1]
NumMFs=3
MF1='low':'trimf',[0 0.16 0.35]
MF2='medium':'trimf',[0.3 0.53 0.68]
MF3='high':'trimf',[0.6 1 1.4]

[Input5]
Name='PORTABILITY'
Range=[0 1]
NumMFs=3
MF1='low':'trimf',[0 0.15 0.35]
MF2='medium':'trimf',[0.32 0.55 0.68]
MF3='high':'trimf',[0.62 0.85 1]

[Output1]
Name='REUSABILITY'
Range=[0 1]
NumMFs=5
MF1='very_low':'trimf',[0 0.1 0.21]
MF2='low':'trimf',[0.19 0.3 0.42]
MF3='medium':'trimf',[0.38 0.5 0.62]
MF4='high':'trimf',[0.59 0.7 0.81]
MF5='very_high':'trimf',[0.78 0.9 1]

[Rules]
1 3 1 1 1, 1 (1) : 1
1 3 1 2 2, 2 (1) : 1
1 3 1 1 3, 1 (1) : 1
2 1 2 3 2, 3 (1) : 1

2 1 2 3 3, 2 (1) : 1

2 2 1 2 3, 2 (1) : 1

2 1 3 3 2, 3 (1) : 1

3 2 2 3 3, 4 (1) : 1

3 1 3 3 3, 5 (1) : 1

3 0 3 3 3, 4 (1) : 1

2 2 1 1 1, 2 (1) : 1

1 1 1 1 3, 1 (1) : 1