Master Thesis
Software Engineering
Thesis no: MSE-2007-19
June 2007

Analyzing Suitability of SysML for System Engineering Applications

Saleem Zubair Ahmad

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Saleem Zubair Ahmad
Address: Folkparksvagan 15:11
37420, Ronneby, Sweden
E-mail: szah06@hotmail.com

University advisor:
Ludwik Kuzniarz
Department of Systems and Software Engineering
Blekinge Institute of Technology School of Engineering
372 25 Ronneby, Sweden

ACKNOWLEDGEMENT

No project could ever get started, without help from a lot of wonderful people. The author wishes to express sincere appreciation to Dr. Ludwik Kuzniarz for his kind assistance in the preparation of this thesis. I would like to thank Dr. Robert Feldt for his technical gaudiness in this thesis. Especially thanks to OMG at www.omg.org for making an incredible platform for modeling and make it possible to access valuable articles for public. In addition, special thanks to all those people whose familiarity with the needs and ideas of this thesis was helpful during the early phase of this undertaking. Thanks also to my mother and friends for their valuable input.

ABSTRACT

During last decade UML have to face different tricky challenges. For instance as a single unified, general purpose modeling language it should offer simple and explicit semantic which can be applicable to wide range of domains. Due to significant shift of focus from software to system "software-centric" attitude of UML has been exposed. So need of certain domain specific language is always there which can address problems of system rather then software only i.e. motivation for SysML. In this thesis SysML is evaluated to analyze its suitability for system engineering applications. A evaluation criteria is established, through which appropriateness of SysML is observed over system development life cycle. The study is conducted by taking case example of real life i.e. automobile product. Results of research not only provide an opportunity to get inside into SysML architecture but also offer an idea of SysML appropriateness for multidisciplinary product development

Keywords: modeling, domain, SysML, suitability, software-centric

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

As a foundation of MBD, the Unified Modeling Language (UML) has been widely adopted by academia and industry since it was introduced by Object Management Group (OMG) in 1995 and became de facto standard within the software engineering in the following years. Although UML was initially developed for software Engineering, it is also suitable for describing processes and products and potentially for System engineering together with other languages such as SysML [3,2].

Actually the Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems by collaboration of several disciplines like hardware, software, and personal. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding towards design activities and system validation while taking into account the complete problem: The elements of the systems engineering regulation are not new. many of the systems engineering concepts, such as a structured approach and traceability of design towards requirements, have been applied to small system development for many years. The major factor which distinguishes Systems engineering is the scale and complexity of the target systems. The huge complex systems are composed of dissimilar "subsystems", which may in turn be large and complex, and require further systems engineering to meet their cost, schedule, and performance objectives. The system complications have been grown up due to increased functionalities and complex relationships between different disciplines. [11, 1]

The International Council of System Engineering (INCOSE) defines the System engineering as *"Systems engineering integrates all the disciplines and specialty groups into a 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"* [11].

The above mention definition could be used to describe the characteristics of the modern industry where several engineering disciplines are integrated with employing different techniques concurrently. System engineers searches a domain specific modeling language to specify a complex systems because System may also include non-software components such as hardware, information, personnel, process and facilities. UML is software bias so UML cannot satisfy the needs about non software systems. Hence need for certain domain specific language like SysML become more obvious because it is believed that SysML may satisfy all needs of complex system including different types of sub-systems. [4]

1.2 Aims and Objectives

The System Modeling Language (SysML^{TM}) is strongly related to UML due to fact that SysML^{TM} is developed through customization of UML for System engineering [1].

In general we easily observed two most important deficiencies in UML from a system engineer's perspective i.e. poor support for data flow, and the difficulty in describing system architecture. System engineers often use data flow concepts, both to describe overall behavior, and to define the key algorithms in system control. The only tool that they currently have in UML is the Activity Diagram, which implements very few of the true dataflow concepts and is aimed at class design, rather than system design. Engineers are also concerned with accurately representing the frequently complex architecture of the total system, something that the current deployment diagram is not representing truly [7] [1].

The aim of research is to observe how the SysML^{TM} can be used to support the development of system engineering application.

The following are aims and objectives of research.

1. Developing of tutorial for SysML for the purpose of understanding and getting inside into language architecture.
2. Selection of appropriate elements through which SysML can be analyzed.

1.3 Research Questions

To achieve aims and objectives, research is focused to explore the effectiveness of SysML for system engineering applications. The main research question is

1. *Why SysML is more suitable then UML for system engineering applications?*

. It should be noted that UML is a general purpose language and can be utilized for modeling system engineering applications, so main research question is designed in such a manner that through this question additional capabilities of SysML can be focused. The phrase "more suitable" in question highlighted fact that UML has some potential for solving complex system engineering problems. Deficiencies of UML in addressing system engineering issues can also be investigated through this question. Actually this question is designed with the aim to investigate and verify the claim of OMG and INCOSE that SysML remove drawbacks of UML in order to address system engineering issues more precisely and accurately.

Before addressing main research question, following sub-questions will be addresses as pre-requisite in order to get inside into details of different involving concepts.

a) *What does make System Engineering more comprehensive then software engineering?*
b) *Why four Pillars are included in SysML Architecture? And what is their contribution?*
c) *What are the key modifications made by SysML in UML structure?*

Through First sub question a comparison between system engineering and software engineering will be established. This study will help in identification of those aspects through which system engineering is considered to be a broader discipline than software engineering. actually this comparison will provide basis for comparing two languages i.e. SysML and UML. Architecture of SysML will explored thorough second sub question. Investigation regarding justification of presence of four pillars i.e. structure, behavior, parametric and requirement will be conducted. How SysML makes changes in exiting structure of UML, this will be investigated by third sub question.

Actually three sub question are formulated in such a way that they not only established solid background for answering main research question but also but also give an opportunity to investigate different aspects of SysML.

1.4 Expected Outcome.

Thesis will contain consolidating knowledge for answering research questions and will make inquiry, documentation and analysis of the SysML including identification of possibilities and obstacles to adopt SysML by considering one or more domain specific system functions.

In short we can expect following in report.

• An understanding and explanation of SysML.
• A justification of effectiveness of SysML for system engineering applications.

1.5 Assumptions

Following two important assumptions have been made regarding this study.

1. Readers of this research document should have good idea of UML.

2. Readers must familiar with some concept about system engineering particularly INCOSE contributions towards system engineering.

1.6 Research Plan

Actually it is a comparative study containing two noteworthy comparisons first; system engineering vs. software engineering and then UML vs. SysML. The research will be conducted in following steps.

- First, a comparison between system engineering and software engineering will be established. The study will be conducted in light of exiting knowledge relating to these two disciplines. Study will also explore different phases of SIMILAR [9, 34] (system engineering process recommended by INCOSE) [11].Basic theme of study is to compare capabilities and domain of these disciplines and decide which is more suitable for current industrial scenario.
- Conduction of investigations regarding architecture of SysML with respect to its four pillars i.e. structure, behavior, requirement and parametric. After this exploration, a comparative study between UML and SysML will be conducted by keeping in view following points.
  1. Which elements are common in UML and SysML?
  2. Which new elements are including in SysML? And why?
  3. Which elements are left out by SysML from UML profile?

  Out of elements which will come out while answering above mentioned questions, three elements will be considered for further study and to address research questions more precisely i.e. Blocks, Parametric, and Requirement. This will be done to make research more precise and time feasible.
- An evaluation criterion will be derived from exiting knowledge [35] to judge the capabilities of SysML for addressing complex industrial problems.
- Selection of appropriate case study [14] to support research result, which will be produced by the application of evaluation criteria on different phases of SIMILAR [9, 34].
- Implementation of evaluation criteria on SysML in context of SIMILAR [9, 34]. Study will aim at, to observe how SysML support system engineering activities in each phase of SIMILAR [9, 34].

1.7 Thesis outline

The thesis is planned out in following chapters.

Chapter 1; This chapter provides introduction and background of study, describing research plan and highlighting some basic assumptions regarding study.

Chapter 2; In this chapter, comparative study regarding system engineering and software engineering has been described. Further more SIMILAR i.e. system engineering process and motivation for adopting system engineering in current industrial scenario is also described.

Chapter 3; It contains structural information regarding SysML architecture, identifying significant elements in SysML structure and also provides justifications for their presence.

Chapter 4; It provides details about the new elements introduced by SysML and also provides idea about some major extensions made by SysML in exiting UML structure.

Chapter 5; This chapter gives details about case study 'rain sensing wiper'; an automobile system. Formulation of evaluation criteria for judgment of effectiveness of SysML is also described in this chapter.

Chapter 6; How SysML is evaluated through newly derived criteria in context of SIMILAR; this is major theme of this chapter.

Chapter 7; This chapter presents conclusions of study and describes some future aspect regarding SysML.

2 A COMPARATIVE STUDY; SYSTEM ENGINEERING VS SOFTWARE ENGINEERING

International Council of System Engineering (INCOSE) [11] adopted the definition of [Rechtin, 2000] for defining system in modern industrial context *"A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behaviour and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected."[9]*

A system is consisting of different individual elements like people, hardware, software, facilities, policies, and documents. The outputs given by system are qualities, properties, characteristics, behaviour, functions and performance. System consists of software and non-software elements [9]. In short, system is a collection of different related components which are trying to achieve one common objective. However it should be noted that computerized system is comprised of elements like hardware, software, people, facilities, and data [10]. So, it become obvious that software is an element of system that has only software intensive tasks.

2.1 System Engineering

The collaboration among several disciplines is an emerging/important trend faced by the modern Industrial circles. The complications in system structures have been grown up due to augmented functionalities and complex relationships between different disciplines. There are so many complex relationships among different functions/components of the system furthermore, during products development different methodologies and different tools are used due to the different design patterns and traditions for each discipline.

Actually the 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 [11]. The elements of the Systems Engineering discipline are not new. Many of the systems engineering concepts, such as a structured approach and traceability of design to requirements, have been applied to small system development for years. What distinguishes systems engineering is the scale and complexity of the target systems. The large complex systems are composed of "subsystems", which may in turn be large and complex, and require further systems engineering to meet their cost, schedule, and technical performance objectives.

Traditionally system engineering is defined *"The set of activities which oversees the development process, ensuring that the product requirements are complete and fully satisfy the customer's needs."[12]*

However keeping in view the characteristics of the modern industry where several engineering disciplines are integrated with employing different techniques concurrently INCOSE define the System engineering as *"Systems Engineering integrates all the disciplines and specialty groups into a 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."* [9]

2.1.1 System Engineering Process

Actually responsibility of system engineering is to establish such a efficient interdisciplinary processes which can easily ensure that customer need are satisfied within prescribed deadline, preserving highest level of quality and in a cost effective manner.

For this purpose INCOSE describe system engineering process which usually called SIMILAR process as shown in Fig 1.[9]

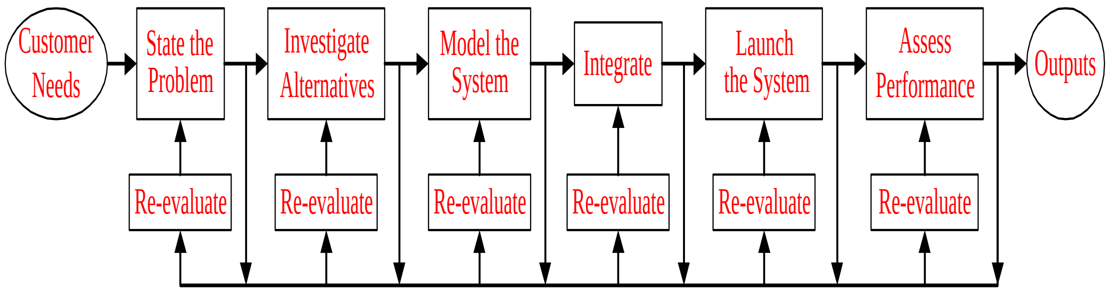


Figure 1. System Engineering Process [9]

The word SIMILAR is actually acronym for following seven different activities which constitute this process. [9]

1. State the problem /problem definition
2. Investigate alternatives / find alternatives to solve problem
3. Modelling the system
4. Integrating the system
5. Launching the system
6. Assessment of performance
7. Re-evaluation

2.1.1.1 State the Problem:

In this stage our major focus what should do with out involving thinking how it will be done. Normally problem statement is more concentrated on functional requirements but we can also take in to account important non-functional constraints. Problem statement should be written in a sense that it can be traceable. Simplification is another important property of problem statement because all kind of stakeholder will interact with this statement so difficult language must be avoided in writing this statement. [9]

2.1.1.2 Investigate Alternatives

Consideration of alternatives is always very important particularly if to different performance goals and want to avoid different mishaps and risk in connection with time and cost. However it should be noted that selection of alternatives should be made in a very careful manner and they must be judge against requirement very strictly. [9]

2.1.1.3 Model the system

Modelling activities will be conducted to capture deeper knowledge about different alternatives. The model make thing more visible and manageable. Engineers may relay on several kind of models like Mathematical models, state machine models, Block Diagrams and object-oriented Models. It should be noted that system engineering is not process but it focus on product as well. So Models of product are also developed to analyze different factors like risk management, trade-offs functionalities. However it should be kept in mind that system engineering process is iterative so Modelling will continue through out life cycle. [9]

2.1.1.4 Integrate

It is clear that system engineering is an interdisciplinary approach so integration among different discipline and sub-system is central issue in system engineering process. Basic theme of integration is to relate different components create overall structure of system. system engineering always emphasis on well-organized integration of different to avoid any mishap during integration engineers must establish feed back loop to get correct any fault as early as possible.[9]

2.1.1.5 Launch the system

In this phase system is thrown into working environment i.e. it allow to do for which it is developed, different alternative are also kept in mind. The aspect of human interaction is very important at this stage because operators (human) can easily assess performance test different types of functionalities offered by system. Some parts of system bought from market (COTS) these gains special attention of engineering at this phase because they always interested in analyzing capabilities of these part in actual environment.[9]

2.1.1.6 Assess performance

Performance assessment is one of the key areas of system engineering, for this purpose different metrics and figures of merit are used. Risk associated with system execution is also assessed. Any factor which can effect performance and decrease customer level of satisfaction is of great importance during performance assessment. In short measurement of different factors is key practice at this phase. [9]

2.1.1.7 Re-evaluate

Basic theme of re-evaluation is establish a continuous feed back loop and make close comparison among out put and requirements of system. Re-evaluation is not a name of particular phase but it is a continuous practice which is distributed through out life cycle. So, system engineering process must be documented and well organized and contain minimum variations. [9]

2.2 Software Engineering

As the time pass by system become more and more complex and large to fulfil needs of modern industry. Majority of systems are software intensive and they can not be designed by team of one or two engineers.

In such scenario importance of software engineering becomes great for modern industry. Software engineering is the process of designing, developing and documenting the software by using technologies from various fields such as computer science, project management, engineering and others. Software engineering use to identify, define, release, and verify the characteristics of resulting software.

These characteristics may comprise of functionality, accessibility, maintainability and other attributes. Software engineering addresses the characteristics of software by addressing design activities and other technical requirements. Software gets ready as result of execution of design and requirements. After design implementation software can be verified to meet all kinds of requirements. [8]

Software engineering can be defined in several many ways, Naur and Randall define software engineering as *"the establishment and use of sound engineering principle in order to obtain economically software that is reliable and work efficiently on real* machines." *[6]*

Boehm give following definition to software engineering "*Software engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer program, procedures, and associated documentations*." *[13]*

It should be noted that Boehm's definition more emphasis is on mathematic, keeping in view more and more formalism involved in engineering activities.

One of the more operational definition giving by FREEMAN and VON "*software engineering is the systematic application of method, tools, and knowledge to achieve stated technical, economic, and human objectives for a software-intensive system."[13]*

IEEE define software engineering in more comprehensive manner "*Software Engineering(1) the application of a systematic ,disciplined, quantifiable approach to the development, operation, and maintenance of software; that is application of engineering to software.(2) the study of approaches as in(1)." [14]*

In general software engineering is considered as a layered technology because we can easily divide software engineering activities into four layers: [14]

1. Process 2.method 3.tools and 4.Quality focus which provide guide line to software engineering regarding TQM (Total Quality Management) frame work. The process layer actually provide foundation for software engineering activities and it holds different technologies layers together and make sure that deadline regarding time, cost will not violate and quality level will be preserved. [14]

Software engineering method provide 'how' information for software development. Method provides guideline line regarding conduction of task like requirement analysis, design patterns, testing and maintenance. In present age software engineering issues become very complex in nature so we have deploy CASE tool for proper conduction of software engineering activities. These make it possible to achieve more quality in lesser time by consuming minimum budgets. [14]

2.3 System Engineering vs. Software Engineering

In the light of above discussion comparison between two disciplines can be established. [14] [13]

- System engineering is comprised of well-organized process through which operational desires and specific requirements are transformed into system.
- Software engineering provides disciplined approach for developing software system by utilizing resources efficiently.
- System engineering hierarchy consist of four views i.e. world view, domain view, element view and detailed view. To established proper business or technology context entire domain will be examined. It is termed as world view, domain comes around when context have been established and engineers start focusing on particular domain of interest. When engineers narrow their focus by paying attention on each element of system individually i.e. elements view.
- According to Pressman [14] software engineering starts from element view, when software element is focused individually. After element view, implementation detail of each element is considered which relate to detailed view.

In light of above mentioned points, it can be easily observed that software engineering activities are conducted within context of system engineering. Actually software engineering and system engineering both are the processes used for the development of some kind of products that fulfils the need of the customers by designing, developing and documenting. Where software engineering is used to create those products only which can perform software intensive tasks. System engineering used to prepare the whole system which includes different parts like hardware, software, documents. In short, software engineering activities occur as a consequence of system engineering activities.

The following table shows comparative capabilities of two disciplines towards different life cycle activities [13].

Where **H:** Good support, **M:** Moderate support and **L:** Low/ Poor Support.

Activities	System Engineering	Software engineering
Alternative design method	H	M
Multidisciplinary Orientation	H	L
User requirement analysis and prototyping	H	H
Criteria-based trade-off analysis	H	L
Derailed software Specification	M	H
Optimal software Production	M	H
Method Integration Planning	M	L
Education and Training	M	M
Artifacts Profiling	M	M
Application range assessment	M	M
Measurements and standards	M	M
Processes and metrics	M	M

Table 1. Comparison between system engineering and software engineering

It is obvious from table.1 that system engineering provide a good support for alternative-selection, multidisciplinary approach, requirement satisfactions and for conducting trade-off. As for as software engineering is concerned, it is strong enough is software intensive areas like software design activities and product optimizations. In the light of current industrial scenarios where systems are comprised of several disciplines, system engineering appears to be better choice for engineers. [13]

2.4 Why SysML?

As a foundation of MBD, the Unified Modelling Language (UML) has been widely adopted by academia and industry since it was introduced by Object Management Group (OMG) in 1995 and became de facto standard within the software Engineering in the following years. According UML reference Manual, UML is Defined as *"Unified modelling language is general purpose visual modelling language that is used to specify, visualize, construct, and document the artefacts of software system."* [15]

UML is general purpose modelling language and is used in understanding, designing, browsing, configuring, and maintaining and control information about software systems. UML includes concepts, notations and guiding principles. UML supported by visual modelling tools and by report writers. UML captures information about static structure and dynamic behaviour of software system. Static structure explains the kinds, implementation and relationships of objects important for system. Dynamic behaviour tells about object's history after short time and communication among objects. [15]

UML is general purpose modelling language which mean it can be use to make design of any particular product. UML is a simple language in understanding and making a software system. The two most important deficiencies in UML from a System Engineer's perspective are the poor support for data flow, and the difficulty in describing system architecture. System engineers often use data flow concepts, both to describe overall behaviour, and to define the key algorithms in system control. The only tool that they currently have in UML is the activity diagram, which implements very few of the true dataflow concepts and is aimed at class design, rather than system design. They are often also concerned with accurately representing the frequently complex architecture of the total system, something that the current UML Deployment Diagram is not fulfilling this purpose comprehensively. [16]

System engineers are always in search of domain specific modelling language to specify a complex system where as software engineers use general purpose modelling language such as UML to design and specify software intensive systems.

System may also include non-software components such as hardware, information, personnel, process and facilities. UML is software bias so UML cannot satisfy the needs about non software systems. Hence need for SysML because SysML satisfy all needs of complex system including different types of parts. [17]

SysML is based on UML 2.0 and now become a standard language for system modelling. This language fulfils all the needs of system engineering [19] [1].

SysML is a graphical modelling language for system engineering developed by Object Management Group (OMG), International council on system engineering (INCOSE) and AP233 (ISO). SysML has a certain team members coming from industry & government, tool vendors, academia and organizations. [1, 18]

OMG definition for SysML is *"standard modelling language for systems engineering to analyze, specify, design, and verify complex systems, intended to enhance systems quality, improve the ability to exchange systems engineering information amongst tools, and help bridge the semantic gap between systems, software, and other engineering disciplines." [20][1]*

SysML provides semantics (meaning) and notations (representation of meaning). SysML supports the requirements, testing, design, verification, and validation of systems. These systems include hardware, software, data, personnel, procedures and facilities. SysML is language independent on methodology and tool. [1, 18]

SysML is a domain specific modelling language. Domain specific modelling is a way of how system design and develop. It uses the domain specific language used to represent the different parts of system. [20]

SysML Language supports model and data exchange. Model based system engineering has certain benefits. Firstly the communication improved between different stockholders like customers, developers, project managers, testers and others. Complexity of system will be managed by model based development. By improving in design quality errors and doubt about system reduces. Risk also reduces by verification and validation process by doing this process at early and ongoing stages of system. [1, 18] [1]

3 SYSML ARCHITECTURE

Actually SysML is evolved to provide simple and effective constructs to address modeling issues of complex system engineering problems. As SysML reuse subset of UML, it seems to be a good approach to describe its architecture with respect to UML as shown by Venn diagram in fig.2, where UML and SysML are represented by two intersecting circles.

Following three areas of concern can be easily extracted from fig.2. [1].

- UML reused by SysML
- SysML extensions to UML
- UML not required by SysML

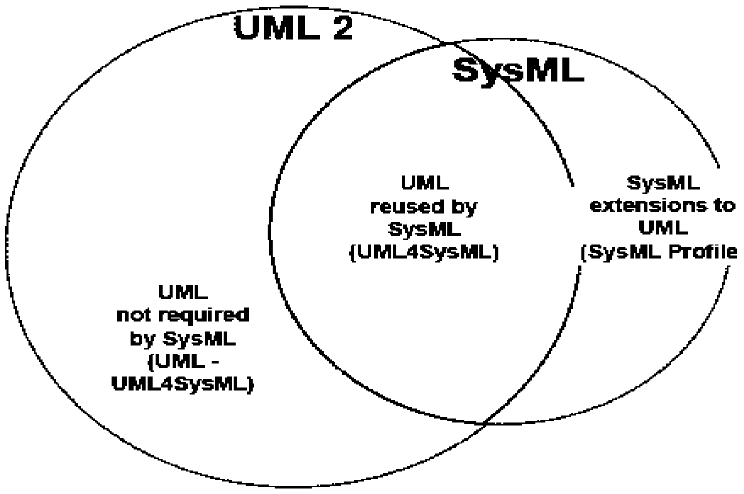

UML 2
SysML
UML reused by SysML (UML4SysML)
SysML extensions to UML (SysML Profile
UML not required by SysML (UML - UML4SysML)

Figure 2. Relationship between UML and SysML. [1]

SysML is comprised of nine standard views/diagrams whereas UML consist of thirteen views/diagrams. Actually SysML retain some diagrams with out modification while a number of diagrams are adopted with modifications. Further more SysML also introduce several new diagrams which are not present in UML. These diagrams are actually considered as extensions made by SysML. SysML diagrams are generally divided into three categories as shown by Fig.3 by keeping in view three areas of concern given by Venn diagram in fig.2 [1].

1. Diagrams that are used same as UML 2.0,
2. Diagrams used with slight modification from UML 2.0.
3. New types of diagrams.

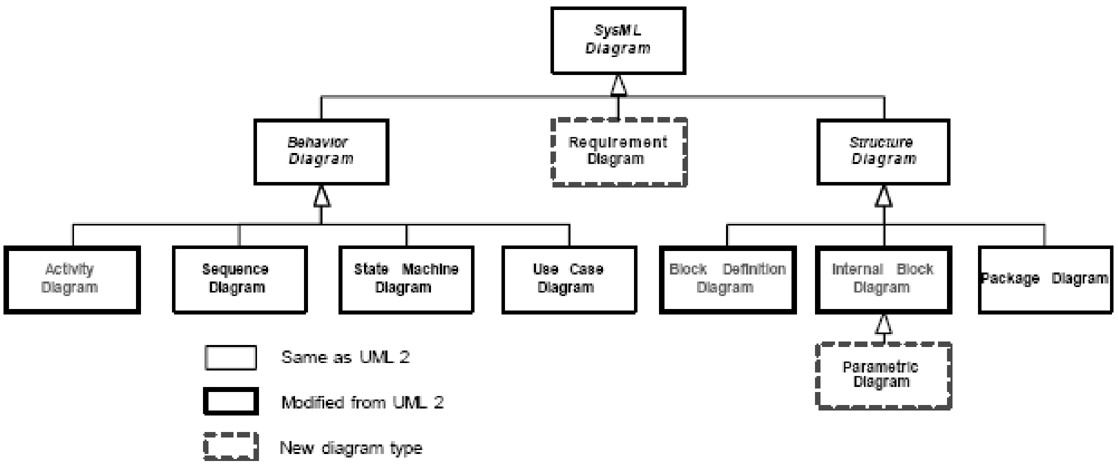


Figure 3 . SysML Diagram Taxonomy [1]

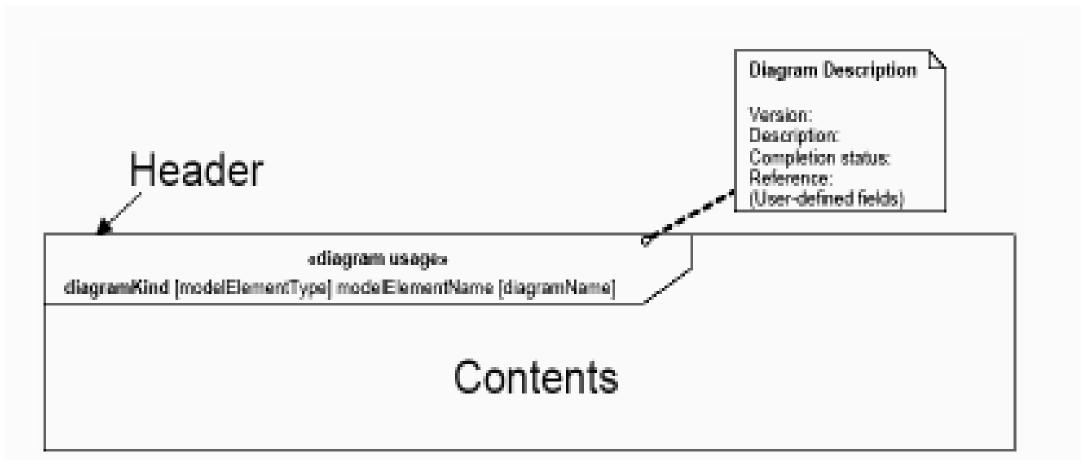


Figure 4. Notation to represent Diagram in SysML [18]

Every SysML diagram represents a model element and it must have diagram frame. Header consist of diagram kind, model element type, model element name and view name/ diagram name in descriptive form [5]. SysML diagrams are divided into four pillars i.e. structure, behavior, requirement, and parametric. Structural pillar provides hierarchical picture of model and gives guideline and regarding application of Block, parts, connectors and ports. Behavioral pillar is comprised of data flows, interactions, activity flow and state machine. This pillar also deals with sequence modeling. Actually these four pillars of SysML make model more consistent and complete and make it possible to observe model from system's point of view .[6]

The requirement pillar brings new concept of requirement modeling and provide formalism in requirement management activities. Parametric pillar brought lot of expressive power to SysML in depicting mathematical equations because modeling of complex mathematical relations/equation is not an easy task for engineers. SysML also introduced several crosscutting concepts like allocations, flows, traceability of requirements and value binding for mathematical expressions. Further more cross cutting constructs provide good support during integration. Components from different disciplines can be integrated with help of these constructs e.g. mapping of software components to appropriate hardware. [23]

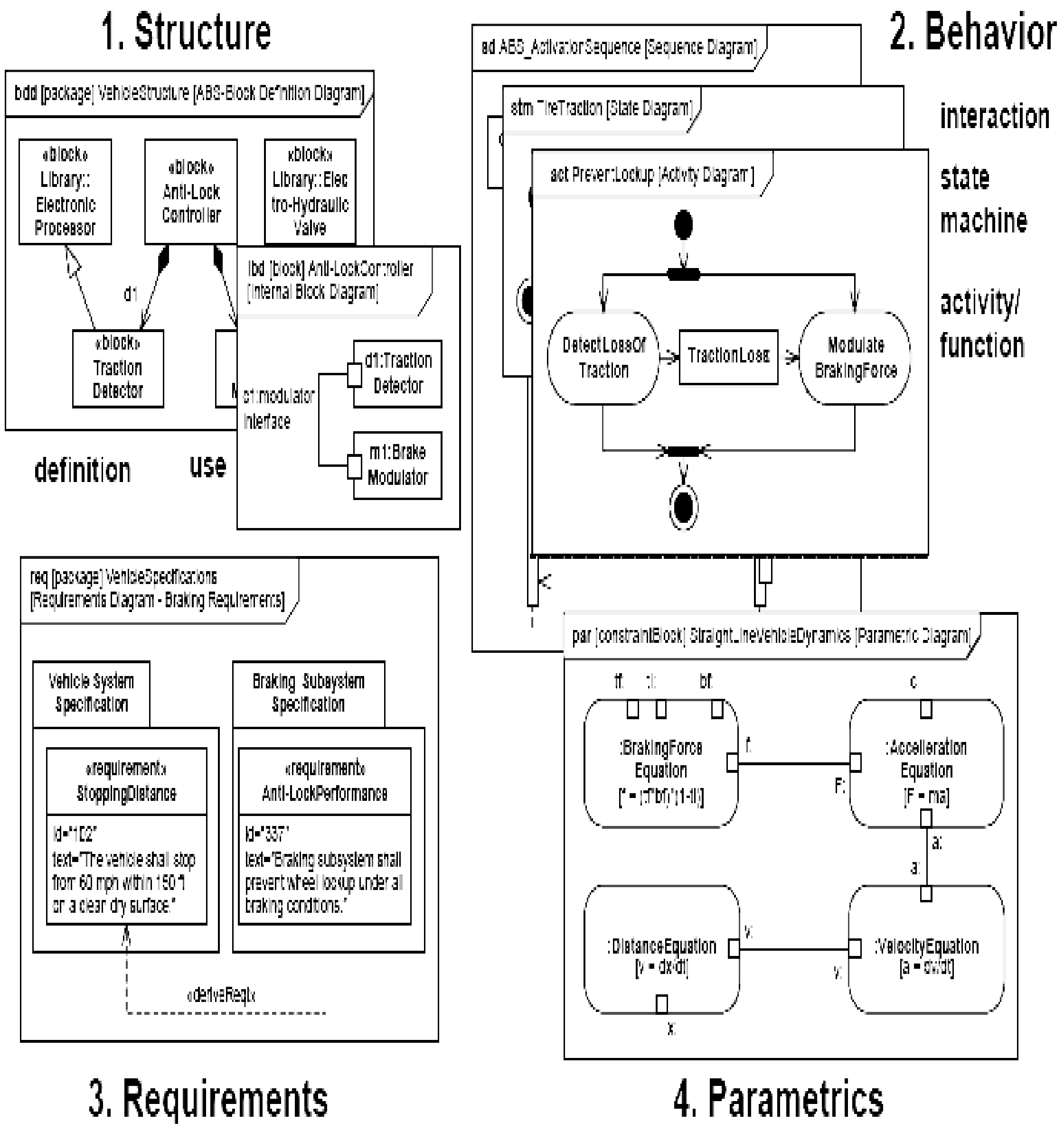

1. Structure
2. Behavior
3. Requirements
4. Parametrics
definition
use
interaction
state machine
activity/ function
bdd [package] VehicleStructure [ABS-Block Definition Diagram]
«block» Library:: Electronic Processor
«block» Anti-Lock Controller
«block» Library::Electro-Hydraulic Valve
«block» Traction Detector
d1
ibd [block] Anti-LockController [Internal Block Diagram]
d1:Traction Detector
m1:Brake Modulator
c1:modulator interface
sd ABS_ActivationSequence [Sequence Diagram]
stm TireTraction [State Diagram]
act PreventLockup [Activity Diagram]
DetectLossOf Traction
TractionLoss
Modulate BrakingForce
req [package] VehicleSpecifications [Requirements Diagram - Braking Requirements]
Vehicle System Specification
Braking Subsystem Specification
«requirement» StoppingDistance
id="102" text="The vehicle shall stop from 60 mph within 150 ft on a clear dry surface."
«requirement» Anti-LockPerformance
id="337" text="Braking subsystem shall prevent wheel lockup under all braking conditions."
«deriveReqt»
par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]
:BrakingForce Equation [f = (tf*bf)*(1-tl)]
:Accelleration Equation [F = ma]
:DistanceEquation [v = dx/dt]
:VelocityEquation [a = dv/dt]
tf: tl: bf: c f: F: a: a: v: v: x:

Figure 5. Four Pillars of SysML [18].

3.1 Structure

In SysML system structure is modeled through block definition diagrams (bdd) and internal block diagrams (ibd). A block definition diagram represents system structural hierarchy and classifications of system/components. The internal block diagram tells about the internal structure of a system in terms of its parts, ports, and connectors and package diagram is used to organize the model. [24]

In SysML, Block is basic unit of system structure. Block can be used to represent software, hardware, facilities, personals and other system elements. System structure is presented in the form of block definition and internal block diagrams. Hierarchy of the system/component is described by block definition diagram and internal structure with respect to its parts. Ports and connectors are described by internal block diagram. [24]

The block definition diagram also represents different types of relationships among blocks like dependencies, generalizations, and associations. Actually block definition diagram represent blocks in terms of relationships, properties and operations. It also has good focus on building system hierarchy. As described earlier that structure of system is described in two types of diagrams i.e. block definition diagram and internal block diagram so it is also possible to represent internal structure of Blocks with help of internal block diagram. This diagram represents internal structure of block in terms of its properties and connectors among properties. Simplest form of properties that block can hold are parts and references to other blocks. There is special class of property which specifies different possible interaction among blocks is called port. The role of block will be represented by property, it should be noted that role of block will be defined by keeping view the role of its internal blocks. Concept of SysML blocks is actually drawn from traditional UML Classes with some modifications like complex forms of association have been avoided in SysML. [1, 24, 30]

3.2 Behavior

Behavioral pillar is comprised of activity diagram, use case diagram, sequence diagram and state machine diagram [24]. Actually Behavior diagrams are drawn from UML 2.0 and SysML has made some enhancement in activity diagram whereas sequence, state machine and use case diagram as same as UML 2.0. The flow of controls, inputs and outputs between different actions is described by activity diagrams. It is now possible to develop timelines corresponding to each activity in SysML, where time model is used to represent constraints and time which makes activity diagram more accurate for certain specific scenario. It should be noted that in SysML activities are classes which are connected by associations. [1, 30]

Usage of system is describes by use case diagram. Actors make interaction with system to achieve a their respective goals. Use case represents functionality achieved by the interactions between system and its actors. Actors are connected with use cases by communicational path represented by associations. Use cases are organized in form of packages and also contain dependency between use cases in packages. Some of important relationships in use case diagrams are “include”, “extend”, “generalization”. In short high level description of functionality is described by use-case diagrams. [1]

State machine diagram is used to represent discrete behavior of system by finite state transition systems. There is no difference between UML state machine and SysML state Machine diagrams. State machine describe dynamic behavior of object over time because object pass through different state during its life time. The major objective of state machine is to track these state with respect to corresponding events [21]. In short it can be said that state machine is graph in which nodes are state and arc are transitions. In general state machine is attached to certain class and represent dynamic behavior of its one instance.

Sequence diagram is used to represent flow of control between different parts of system or Actors and system. Sequence diagram represents sending and receiving of messages between interacting entities called lifelines and time is represented by vertical axis. This diagram represents complex interactions. In short we can say that sequence diagram is two-dimensional chart. Messages are represented on horizontal axis and time is represented on vertical axis. [21]

3.3 Requirement

The support of requirement modeling is one of the major advancement made by SysML because as for as UML is concerned there is no explicit support of requirement representation in UML and considered as one of significant shortcoming in UML.[24,1]

SysML provides modeling constructs through which text based requirement can be modeled. In SysML requirement is represented as stereotype of class. A set of stereotyped dependencies is also available in requirement diagram. The “verify dependency” is used to establish link between test case and requirement it verifies. Further more it is also possible to show that model element is refinement of some textual requirement with “refine dependency”. “copy relationship” indicates reuse of requirements in different hierarchies. It is also possible to track derived requirements through derive stereotype. It is always a problem to model whether design elements satisfy requirement or not, SysML solve this difficulty by providing “satisfy dependency”. Requirement diagram also provides containment relationship, this relationship help the designer in decomposing requirement into it’s constitute. [1, 24]

In short requirement diagram gather requirements and builds requirement hierarchies. Different types of relationship will serve as glue to hold requirements together in hierarchies. It is also a good practice to relate important properties like verification with requirements. Actually requirement diagram establishes a bridge between requirement management tool and SysML Models. [1, 24]

3.4 Parametric

Parametric pillar consists of parametric diagrams which is actually a specialized version of internal block diagram. One of major objective of SysML is to solve complex engineering issues and to provide effective solutions.

Parametric diagram in SysML provides a very useful way to model complex mathematical problems. These diagrams are actually used to model the relationship in the form of mathematical or logical expression or constraints which act on the certain set of values. The major element of this diagram is constraint block. Actually constraint block describe a set of parameters and respective constraints on these parameters. The constraints block incorporates constraints such as {F=m*a} and parameters of constraints like F, m and a. Constraint can be reusable because constraints blocks define constraints in generic form i.e. constrains can be applied to several context after their definition. The most important task carry out by these diagrams is to relate the engineering analysis with design models where engineering analysis deals with the different characteristics of the system like performances, reliabilities. Parametric diagram also supports reusability; it is possible to reuse the mathematical equations and logical expression in the SysML. [1, 24]

In short, parametric diagram provide efficient usage of constraints blocks which contain mechanism for engineering analysis. Constraints block binds different constraint parameters. It should be noted that time can be Modeled as a property in Parametric diagram. These diagrams also play an important role in trade-off analysis. [1]

3.5 Cross-Cutting Constructs

Cross-cutting constructs play a significant role in SysML architecture. Actually these construct map one element to another. These constructs may take the form of allocations, requirements, and parametric. Most important cross-cutting construct is allocations which define a basic allocation relationship that will be used to allocate a set of model elements to another, such as allocation of model elements from behavior to structure constructs or allocating from logical to physical components or from software to hardware.

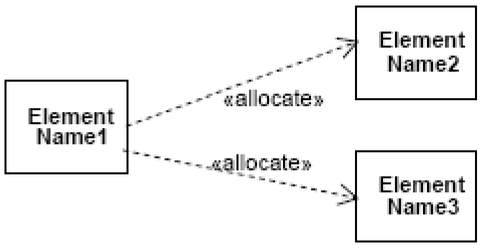


Figure 6. allocation relationship [18]

Actually allocation relationship provides an effective way to navigate model by creating cross relationship. Further more it also ensures that various part of model are appropriately integrated. The major concern of cross-cutting constructs is to support activities that will be conducted across the different views, and may be addressed by all or disparate parts of the model. [24, 18]

4 UML VS SYSML

In recent time UML has become a de-facto standard of software industry. The language is widely adopted for analyzing and designing software. It is common thinking that UML is software centric and can not be used for system engineering but only suitable for software engineering. However UML has been used for system engineering for many years. [26]

However it should be noted that UML is facing certain challenges relating to system engineering. These challenges include representation of inputs and out puts, continuous systems, physical structures and system characteristics like reliability, safety, performance and parametric relationships. [25]

Further more, it is not appropriate to hold idea that UML and SysML are two separate languages. In fact SysML provides a set of useful extension to UML to support system engineering activities. Although OMG has made several enhancement in UML 2.0 to support system engineering like UML 2.0 has been improved for specifying large and complex architectures and it became easy to decompose complex architectures by applying UML 2.0 structured classifiers, activity diagram partitions is one of major enhancements. The capability of UML 2.0 to integrate structures has been enhanced by cross cutting functionalities. Actually through this functionality model and behaviors are integrated. Further more UML 2.0 also make enhancement in activity diagrams. [25]

No doubt these developments makes UML quite suitable for system engineering but need for certain customize language is always there. For this purpose Systems Engineering Domain Special Interest Group (OMG SE DSIG) [22] and INCOSE[11] work together to customize UML for special needs of system engineers.

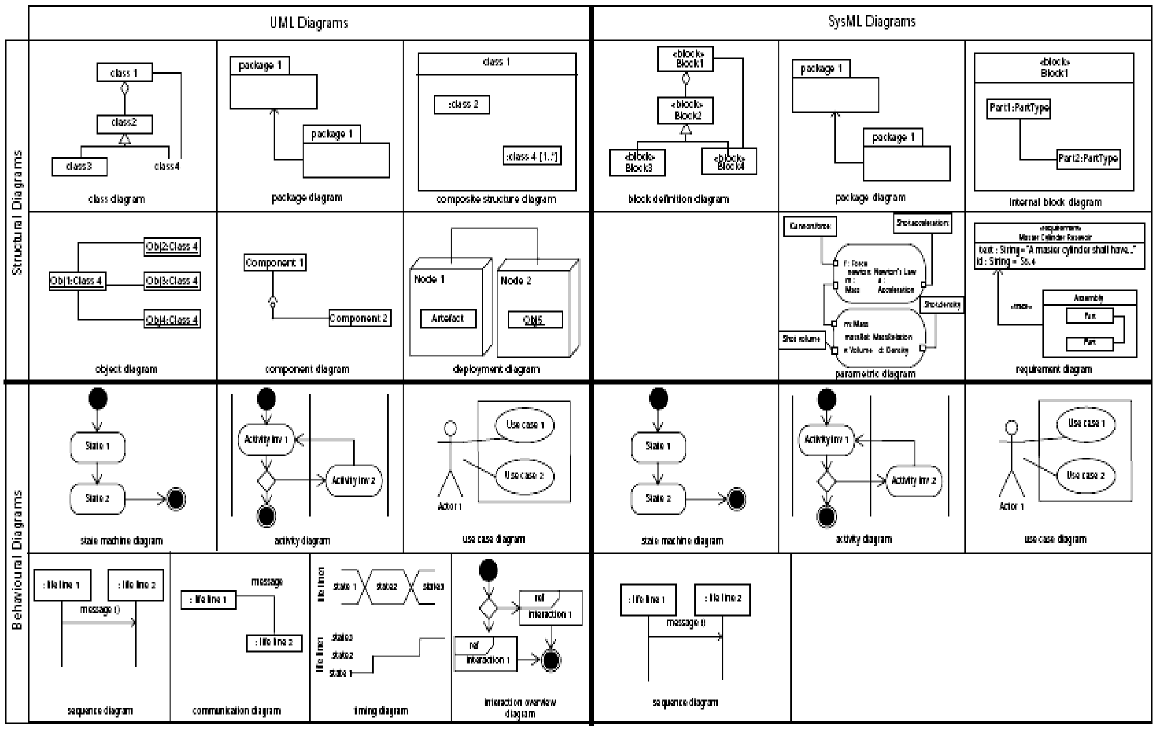


Figure7. Comparison of UML and SysML diagrams [26]

Fig.7 shows that UML provide 13 diagrams i.e. for modeling structural views six diagram are allocated and remaining seven diagram are allocated for representing behavioral aspects. Actually SysML customize UML profile, it introduce two new diagram i.e. for modeling real-world constraints SysML provides parametric diagram and requirement diagram for representing requirement and their corresponding attributes. Further more SysML has tailored some of UML diagrams i.e. block diagram actually an enhanced form of UML class diagram. Similarly UML composite structure diagram is modified as internal block diagram. SysML also extended some functionalities of UML activity diagram. [1]

The idea of customization looks good in sense because according to OMG goal of SysML is to provide "*standard modeling language for systems engineering to analyze, specify, design, and verify complex systems, intended to enhance systems quality, improve the ability to exchange systems engineering information amongst tools, and help bridge the semantic gap between systems, software, and other engineering disciplines". [1, 27]*

Actually when scope of modeling activities become wider i.e. from software to system, engineers have to consider all elements of system including hardware, software, data, personnel, procedures, and facilities. To address modeling issues with respect to all system elements customization of UML becomes an important issue for engineers. [1][27]

It should be noted that many system engineers believed that UML is quite efficient and flexible that it can support extensions made by SysML group to make it possible that UML support system engineering activities more efficiently.

Assertion	True/False	Comment
UML is only suitable for modeling software systems	False	UML has been successfully used for systems engineering and process modeling for many years
SysML and UML are based on the same key concepts	True	These concepts hold true for SysML just as much as for UML
SysML provides systems engineers with a full set of modeling constructs	False	Issues such as timing and deployment are not directly supported by SysML
SysML provides useful system engineering extensions to the UML	True	Flow ports, item flows and parametric diagrams are particularly useful extensions

Table 2. Comments Table [26]

Table2. represents relationship between UML and SysML and tries to answer some misapprehension which the people have in their minds.

4.1 Elements Introduced by SysML

The major challenge faced by SysML group at OMG and INCOSE was to make changes in UML 2.0 but they had to make sure that only extremely necessary changes were made. They tried their best to use as much UML as they can. In the beginning lot of changes had been suggested but SysML group recommended following new concepts

4.1.1 Requirements Diagram

Requirements are foundation stone for every system development. In UML there is no direct support of requirement engineering to make-up this deficiency SysML introduced support of requirement engineering in a sense that engineer not only build requirement model but also relate requirement to actual design elements and test procedures. Use case model in UML can only develop an understanding of system but can not help greatly towards traceability of requirements because through use case, requirements are traced to use case only not to design elements. In UML *note* is used to write comments but it is not considered as model element i.e. designer may exclude these comments from model if they want and there is always a chance that some valuable comments may be lost. To avoid this difficulty SysML requirement diagram provides concept of *rational,* which is considered as model element. In fact *rational* provides space for writing comments against requirements, so that it becomes traceable to design element and test case. *Rational* information actually makes sure that design decisions should also become part of requirement diagram. The rational information may also be helpful for analyzing change impact on requirements. SysML requirement diagram provide a great support for traceability of requirements, and define a trace dependency for this purpose, through this relationship it is possible to relate derived requirements to source requirements. This diagram also play very significant role in solving different complexity problem in requirement, it is possible to decompose requirements through containment relationship just as we can develop containment relationship in UML class diagram. It is always a problem for modeler to link requirement with design elements; this difficulty is by providing satisfaction dependency. It is also possible to relate requirement with associated test cases by means of verification dependency. [31][28]

According to Hause et al [29] one of the strengths of UML is that it is more than just a set of diagrams but there exit an underlying model, which supports the diagrams. In a similar manner requirement diagram also has requirement model behind it. The SysML proposal states "*The requirements model describes the SysML support for describing textual requirements and relating them to the specification, analysis models, design models, etc. A requirement represents the behavior, structure, and other properties that a system, component, or other model element must satisfy*". [29]

However it should be noted that SysML depicts requirement as requirement stereotype of UML class [7].In short, modeling of requirements specifications is very much possible in SysML by keeping in tact different aspects regarding requirements like priority, traceability. In fact only basic features of requirements can be modeled directly, for modeling certain specialized requirements like performance, storage, design constraints designer have to relay on stereotypes.

Notation of requirement diagram in SysML is attached at appendix I [1].

4.1.2 Parametric Diagram

One of the significant advancement made by SysML is inclusion of parametric diagram. Parametric diagram use constraints block as constraints properties. In fact a constraint blocks package expression of constraints. These constraints are packed up in such a way that they can be reusable for other constraints blocks. [1]

Actually through constraints blocks SysML provide mechanism for conducting engineering analysis such as performance and reliability. With the help of constraints block it become possible to establish network of constraints along with their respective mathematical expressions. Parametric diagram also has capability to bind parameters of constraints actually all properties of constraints blocks are defined in term of its parameters. In short we can say that parametric diagram is a restricted form of internal block diagram that represent constraints block along with their respective constraints. The keyword <<constraints>> is used in block definition to show that it is a constraints block. Constraints compartment of block definition will contain expression that specify constraints. However it should be noted that SysML does not provide any particular language for writing Mathematical expressions. [1]

Notation of parametric diagram in SysML is attached at appendix II [1].

4.1.3 Allocations

Concept of allocation is one of new area included by SysML. Actually allocation in SysML is an abstract form of deployment in UML [3]. Actually allocation is a design time relationship between model elements which is used for mapping from source to target. It provides generalized capability to map one element to another. The designer can enforce consistency between different parts of model elements by means of allocations. In SysML allocation is represented by allocate dependency. [4]

In general allocation establishes relation between structure and behavior. Following types of allocations are available in SysML. [1]

4.1.3.1 Behavioral Allocations

Actually behavioral allocations narrate to typical system engineering concept "segregating form from function". According to this concept there should exist an independent model of function i.e. behavior and form i.e. structure and independent mechanism for mapping. However it should be noted that this concept is not supported by object oriented approach. Although according to system engineering point of view this seems to be important idea and is very helpful in solving large scale complex problems. [1]

4.1.3.2 Flow Allocation

This type of allocation exclusively maps flows in functional system illustration to flows in structural system depiction. However it should be noted that flow between activities may be control flow or object flow. [1]

4.1.3.3 Structural allocations

For this type of allocation it is necessary to develop two separate representations of system i.e. logical and physical. Engineers should have definite plan for mapping between these two depictions of system. [1]

4.2 Extensions

4.2.1 Activity Diagram

Actually Activity diagram is used to express the flow of controls, inputs and outputs among different actions. SysML provide some useful extensions to UML2.0 activity diagram which is shaped up as SysML activity diagram. Here are some of extensions that made by SysML. [1]

SysML provide three basic concepts for activities i.e. definition, usage and instance. Definition basically deals with description of activities independent of thinking how this activity is used by other activity. Further more SysML also consider how activity is used in context of other activities, this is called usage concept. When it is necessary to consider execution of an activity, it is dealt by SysML concept called instance. [32]

First major enhancement made by SysML is extension of Control in Activity diagram. SysML broaden the concept of control, for starting and disabling of actions which are already executing while controls in UML are only able to start actions. Capability of disabling actions is because SysML indulge controls values as data. In SysML control operator treated control as data and control value is an input or output of control operator. [1]

Further more SysML confine the rate at which entities flow along edges in an activity. This restriction is applied on both types of rates i.e. discrete and continuous. Inclusion of stereotype <<overwrite>> and <<no buffer>> is very significant because these extensions make sure that up to date information is available for actions because old values will not be kept in object nodes. SysML also introduce concept of probabilities in activity diagrams. [1]

Following are some important stereotypes in activity diagrams.

4.2.1.1 Probability:

<<probability>> this is one new concept introduced by SysML and it is applied to activity edges or on the output parameter set and edges have decision nodes or object nodes as sources.[1]

<<probability>> stereotype provide an expression for the probability that the edge will traversed when this stereotype is applied to edges of decision node and object node. It should be noted that when this stereotype is applied to activity edge then this stereotype is become applicable to all other edges coming out of same source. [10]

4.2.1.2 Rate:

<< rate >> number of objects and values for each time interval. When these values or objects will pass from particular edge is specified by rate stereotype. Actually time interval is the rate from which object leaves the source node and reaches to the target

node. Parameter must be streaming for application of this stereotype and rate of parameter must be less than or equal to the rate of edges Matching to parameter. It should be noted that Flow of objects and values may be continuous or discrete. [1]

4.2.1.3 Optional:

The lower multiplicity is zero only when <<optional>> stereotype is applied on parameters. Constraint for this stereotype is that when this stereotype applies to parameters then lower multiplicity must be equal to zero otherwise multiplicity greater than zero, where lower multiplicity zero means that parameter is not required value for activity to begin execution. While the lower multiplicity greater than zero means parameter required value. [1]

4.2.1.4 Continuous:

<< Continuous >> stereotype is actually a special case of rate and it represent as a continuous flow like flow of water. There is no restriction on the flow rate. [1]

4.2.1.5 Discrete:

<<Discrete>> is a special case of <<rate>> stereotype. In discrete rate, time between two items is non-zero. It should be noted that <<Discrete>> and <<continuous>> stereotypes cannot be applied at the same time on same elements [1].

4.2.1.6 Control Operator:

<<Control Operator>> stereotype is applied both on behavior and operations and is an arbitrarily complex logical operator that can be used to unable and disable actions. It should be noted that when this stereotype is applied to behavior, the behavior then takes control values as input or it may provides control values as output and it treats Control as data. An important consideration in this regard is that when this stereotype is not applied then behavior or operations may not have any parameter typed by control value and if it is applied then behavior or operation must have at least one parameter typed by control value. [1]

4.2.1.7 No Buffer:

This stereotype is applied to object nodes is used to prevent buffer overrun and is used with fast and continuously flowing data values. When the token reach at the out going edges then these edges may refuse to accept and these token may be discarded. <<nobuffer>> stereotype when applied it indicates what happens with token in case of accepting by object node. Token is refused when it arrive at object node in case when stereotype is not applied. [1]

4.2.1.8 Overwrite:

This stereotype is typically used on an input pin. Full object node means that node which holds as many tokens as they allowed by its upper bound. <<overwrite>> stereotype is also applied on object node. As token arrive at full object node it substitute those that are already there. If upper bound is one means it guarantees old data is overridden at an input pin. And if it is greater than one then it would be last according to ordering kind for node. [1].

Notation of Activity diagram in SysML is attached at Appendix V [1].

4.2.2 Block Diagrams

These diagrams are actually providing extension to UML class diagrams. SysML blocks are an enhanced form of UML classes by providing capability of reusability and nesting of connectors. Actually Block is basic unit of structure of SysML and it can represent software, hardware, facilities, personals and other system elements. According to SysML block diagram concept System structure is presented in the form of block definition and internal block diagrams. Hierarchy of the system/component is described by block definition diagram and internal structure with respect to its parts, ports; connectors are described by internal block diagram. One of important structural feature of block is called property which represents role or usage of its in context of its enclosing block. [1]

4.2.2.1 Labeled compartments

A block in SysML can be partitioned into multiple compartments. Each compartment is to show that what kind of information it can hold. Actually compartments in blocks are used to partition the features. However it should be noted that SysML define some standard compartments. There is no specific order to represent compartment i.e. compartments may be organized in any order [1]. Following are some important compartments in SysML Blocks [1]:-

4.2.2.2 Constraints compartment

This compartment holds one or more constraint imposed on block. The constraint can be written in simple text-based notations or using brace, which is syntax to write constraints in UML. Constraint property also declared within this compartment. The use of constraint is optional i.e. it is totally decision of designer to include this compartment or not.[10]

4.2.2.3 Namespace compartment

This compartment is used to declare namespace of block. A label *namespace* will appear in definition of block that is defined in namespace of other block. This feature is considered to be useful in a sense that same block may occur several time in same diagram.[1]

4.2.2.4 Structure compartment

Structure compartment will represent connector and other internal structural elements. A label structure is used to show this compartment. As block can appear several times in same diagram, so it is necessary to represent this compartment separately. Normally this compartment constrains graphical elements so wider compartment is required for structure. [1]

4.2.3 Ports and Flows

According to OMG SysML specifications port is defined as "*A port is an interaction point between a block or part and its environment that is connected with other ports via connectors*". [1]

Port provides the block, capability of interaction with its environments or to other blocks. One of advantages to have port on block is make block reusable, and to have definite and independent mechanism for communication among blocks. It should be noted that block can own its port; therefore port can be considered as part of block definition. [1]

Following types of ports are provided by SysML. [1]

4.2.3.1 Standard Ports

Standard ports having a very simple concept of communication i.e. it provides services of blocks to environment as well as if block requires some thing from environment then standard port come in to play. Concept of standard port is very useful for service oriented architecture and peer-to-peer communications where it is much need to have bi-directional flow of data. [1]

4.2.3.2 Flow Ports

Actually flow port is used to specify the input and output that can enter and leave blocks. In geranial flow port are for asynchronous communication. In short we can that flow port is used to represent item that can flow between block and its environment.[1]

4.2.3.3 Item Flows

Important distinctions between flow ports and item flows is that flow port specifies what can flow in and out of blocks where as item flows specify what does flow among blocks/or part and their respective associations.[1]

Notations of block diagram in SysML is attached at appendix III, IV [1].

4.2.3.4 Outcome

It has been observed that SysML is not a new language but it is only customization of UML to make it favorable for system engineering activities. SysML group at OMG tried it best to conduct only modification in UML which is entirely necessary and tried to retain maximum elements of UML in SysML.

SysML retain state machine, interaction and use case from UML 2.0 with out any modifications where as activity diagram and block diagram in SysML are enhanced version of activity diagram and composite structure diagram of UML 2.0.[33]

However it should be noted that in UML 2.0 Interactions are supported by four diagrams names; sequence diagram, communication diagram, interaction overview diagram and timing diagram. SysML only includes sequence diagram and excludes other three. [1]

It has been aim of SysML designers to keep language as simple as they can for this several complex form of UML relationships are avoided in SysML, like n-ary association and qualified association are excluded from SysML. Navigation is also simplified by removing arrowheads on both ends, how ever it should be noted that in case of unidirectional association SysML also have arrow heads as UML. Generalization relationship between associations is also dropped from SysML. All these efforts have been made to make language as simple as possible.[1]

5 DEVELOPMENT OF EVALUATION CRITERION

At present time complexity of system is increasing continuously by introduction of new technologies. Although system engineering try to address these upcoming issues particularly in areas of technology and complexity management but it seems to difficult by relying on exiting approaches. Actually SysML provides new way out to address different concerns in system engineering process.

Basic theme of this research is to analyze suitability of SysML for system engineering applications. According to Friedenthal and Roger Burkhart evaluation criteria for the systems modelling language should comprise of following points. [35]

- **Simple.** All stakeholders should not feel any difficulty in understanding language.
- **Explicit.** The semantics of language should be definite and it contains very straightforward set of notations.
- **Precise.** Semantics of language should be easily translated in to mathematical representations. This property of modelling language helps engineers to conduct verification and validation activities more accurately.
- **Complete.** The language should support verity of interest and activities through out system development life cycle.
- **Scalable.** Concept like generalisation/specialization, dependency, compositions should be supported by language to address problem in modelling scalable solutions for complex system.
- **Adaptable.** Modeling Problems of different domains like telecom, automotive should be addressed equally by language. Further more it is extendable.
- **Evolvable.** Language should be flexible that it can easily adopt changes, it also compatible with previous versions.
- **Data Interchange.** There should exist a support of AP-233 and OMG XMI format to exchange semantic information among different tools. Further more language has ability to interchange diagram which in its turn support model exchange among different tools.
- **Independent.** there should exist a support of industry standard systems engineering technical processes, EIA 632 and ISO/IEC 15288 and should not depend on any specific process

5.1 Evaluation Criteria for SysML

Keeping in view evaluation criteria of Friedenthal and Burkhart [35] for analyzing effectiveness of modeling language. A criteria for evaluating SysML has been driven which consists of following points

- **Simplicity: -** Stakeholders should not face any problem in practicing SysML.
- **Explicitness: -** SysML should offer very clear-cut set of notations.

- **Preciseness: -** SysML should support mathematical representations.

- **Completeness:-**SysML should support verity of interest and activities through out SIMILAR [9, 34].

- **Scalability: -** Different modeling concept like generalization, dependency, elaborations and refinements should be supported by SysML to address problems of modeling scalable solutions for complex systems.

- **Flexibility: -** SysML should support change management issues.

Points like evolvable, data interchange and independent are left out from original criteria [35]. As SysML is new language which is gaining maturity with the passage of time. For this reason aspect like 'evolvable' which deal with future changes can be ignored. 'data interchange' point deals with providing some standardized protocol through which data among different tools can be interchanged, at present not much tool support is available for SysML, so this point is not included in derived criteria. However according to OMG SysML has support of AP-233 and OMG XMI for interchanging data among different tools. SysML is strongly based on UML and it is evident that UML is process independent. So 'independent' aspect is not incorporated in derived criteria. It should be noted that a new point called 'flexibility' has been added in derived criteria to address change management issues.

Above mentioned derived criteria is imposed on each phase of SIMILAR [9, 34] to analysis suitability of SysML for system engineering applications. For this purpose we consider case study of "*IBM Rain Sensing System*" [14] and investigate how SysML can support SIMILAR during the development of this system. Where SIMILAR [9, 34] represent system's life cycle in context of system engineering. It comprise of seven tasks i.e. State the problem, Investigate alternatives, Model the system, Integrate, Launch the system, A**s**sess performance, and Re-evaluate[9,34].

After implementation of above mentioned evaluation criteria on SIMILAR [9, 34], summarized result will be formulated in form of "Evaluation Table". The extent to which SysML can support system engineering activities at each phase is denoted by three support levels; H, M and L where H stands for high, M for Moderate and L for Low. These levels are defined in light of study of different industrial case studies, observation and experiences. Results of study will be formulated by considering following two points.

1. To investigate how SysML can support different activities at each phase of SIMILAR in light of above mentioned evaluation criteria.

2. To explore possible improvements in system engineering activities by introduction of SysML.

5.2 IBM Rain Sensing Wiper (RSW); Case Study

At IMB research division *Laurent.B* considered an embedded system called *Rain Sensing Wiper RSW* [33] to demonstrate capabilities of SysML in product life cycle. The motivation for selecting this case study for analyzing SysML capabilities is that IBM playing pivotal in development of modeling tools. Further more case study is base on system through which hardware, software aspect can be considered comprehensively.

Actually RSW belongs to automobile industry. Important characteristic of RSW is that various engineering discipline have been involved in designing this system. The major objective of system is to clean windshield. When some liquid is detected on windshield, system starts working without any involvement of user.

System consists of three types of components;

1. Software i.e. algorithms, which controls activities of hardware.
2. Electronic control unit, which execute software.
3. Sensor, which detects droplets on screen.

Engineers will integrate these components to establish overall structure of system. The motivation for selecting RSW system is that verity of disciplines are involving in system structure, it can provide a good platform for analyzing capabilities of SysML for system engineering applications. Actually system is based on two parameters.

- Windshield optical and geometric specifications, specially its thickness and its optical indexes.
- Operational ranges of sensors

To make system more flexible it is necessary to specify certain ranges for windshield and sensor. This feature permits engineers to make important design choices. However it should be noted that for best performance there should exit great compatibility between values of windshield properties and sensor. [33]

6 APPLICATION OF CRITERIA ON SIMILAR

6.1 State the Problem

In fact, there is no direct support of requirements in UML. Use case Model is only available assistance in UML for requirement elicitation and earlier problem description is. However it should be noted that some time it becomes useful to draw sequence diagrams, state machine diagram and activity diagrams in conjunction with use case diagram to elaborate problem

Use case diagram for RSW can be developed in following manner. The Fig.8 shows three actors, driver, who can manually disable the system, maintenance, person responsible for repairing and car electrical system, which activate RSW in car. <<Include>> dependency is used to decompose use cases. It is obvious from fig.8 that use case diagram can only provide high level picture of requirements, it seems to appropriate for high level requirements analysis but for details requirement analysis engineers have to replay on traditional methods. [33]

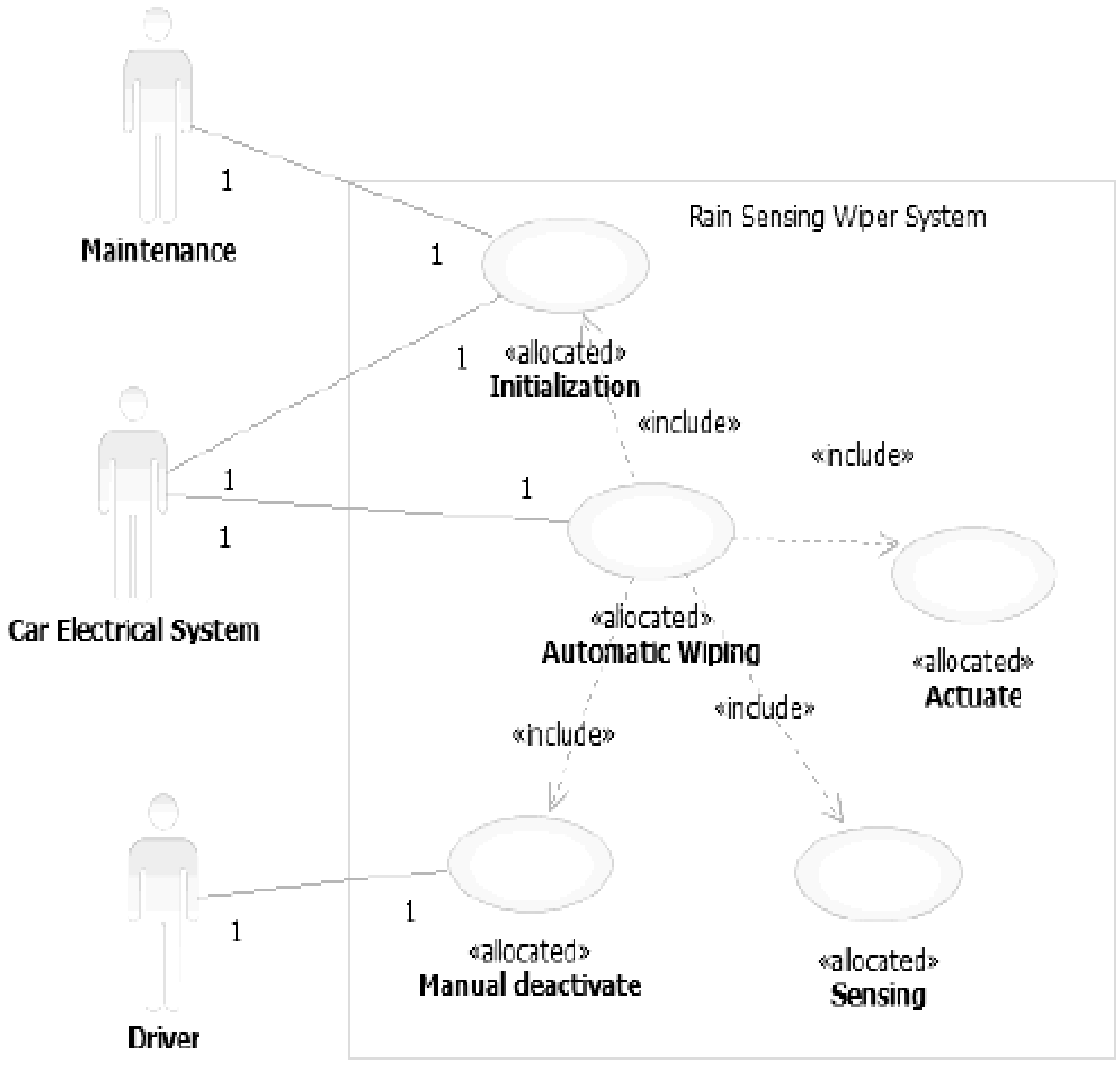

Rain Sensing Wiper System
Maintenance
Car Electrical System
Driver
1
«allocated» Initialization
«include»
«allocated» Automatic Wiping
«allocated» Actuate
«allocated» Manual deactivate
«allocated» Sensing

Figure 8. Use case diagram for RSW. [33]

Requirement diagram in SysML provide a solution for above mentioned problems, now it become possible to elicit requirements in a clear cut manner. Further more taxonomic relationship can be among requirements, relationship in requirement diagram having different types of stereotype like <<trace>>, <<include>>, <<assign >> [1].

One of the major contributions of requirement diagram is the providence of model-level traceability of requirements [6]. Before inclusion of requirement system engineers were using different notations and tools. Due to this inconsistency several problems like reduced traceability, information transcription errors among tools may occur. [19]

System engineering always require a well defined set of requirement and emphasis that measurable processes should be implemented through out life cycle. Industries like automotive, telecommunications and aerospace where system engineering application play dominant role always mandate the idea of well organized requirement management. Traditionally system engineer prepare some structured documents or spreadsheets for establishing requirement definition. When a well defined and agreed upon requirement definition come on to surface, it is communicated to all team members. Now by having SysML, it become quite easy for system engineers to model requirement in a disciplined manner and also structure them with help of several relationships available in requirement diagram. From system engineering point of view it is very important to have requirement traceability so that requirements can be verified and validated at any time. [37]

The IEEE Standard Glossary of Software Engineering Terminology defines traceability as follows:
"The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match; (2) The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies". [38]

Gotel and Finkelstein define requirements traceability as *"The ability to describe and follow the life of a requirement, in both a forward and backward direction; i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases".* [38]

SysML stereotype <<trace>> made it possible for requirements to remains visible through out life cycle. However it is worth while to mention that traceability should be handled very carefully across life cycle, so that changes can be managed dynamically [37].Now system engineers can utilize requirement diagram for taking high level design decisions and also for assessment and verification of different design alternatives [39].
Actually SysML requirements diagram provide system engineers following three important services. [I1d]

- **Transformation of text to Model.**

Every systems development process starts with requirements gathering, at starting point requirements are organized in the form documents contain some important points, and requirements are structured under these heads. SysML requirements diagrams provide a disciplined approach to convert text documents in requirement model. [40]
Now it is possible in SysML to associate several attribute with requirements, these attributes actually specify "shall" statement corresponding to each requirement [1, 9].

- **Derivation**

The SysML term requirements derivation as requirements trace. According to [Cantor, 2003] some assumptions about system design are involved in each stage of derivation. Actually a derived requirement can use trace relationship for establishing relationship with one or more source requirements. [41, 40]

- **Links to System Design**

One of important feature provided by SysML for requirement Modeling is capturing of grounds for any derivation or design alternative. This feature is called *rational.* In general "satisfy" relationship is used to depict how design elements satisfy its corresponding requirement. The *rational* is attached to satisfy dependency, containing reasons, logics and other consideration which have been made in satisfying requirement. Actually *rational* is an extension of comments in UML. Through *rational,* it becomes very easy for designer to capture design decisions. It should be noted that *rational* can be attached to any model element. [1]

In the light of above mentioned discussion requirements diagram for RSW can be modeled with the help of SysML as shown in Fig9. The diagram shows both functional and non-functional requirements. Complex requirements can be easily decomposed in to its sub-requirement by using "composite relationship". For example in fig.9 requirement *Automatic Wiping* contains several sub-requirements, it is denoted by cross enclosed by circles. However it should be noted that if parent requirement is deleted all of its child requirements will deleted like if requirement *Automatic Wiping* is deleted all of its sub-requirements will be erased. [1]
Fig.9 also shows "refine relationship", according to this relationship requirement can be refined with help of other requirements. For example in RSW requirement can be refined further by selecting different speeds i.e. slow, medium, fast. Another significant relationship in fig.9 is "drive relationship", for example *Automatic Enablement* is derived from *Core function* requirement. [33]

Another relationship between requirements is *Refine*. An example of requirement refinement is shown in Figure 9. The requirement on speed actuation is refined by the possible selection for speed (slow, medium or fast.) Lastly, a generic *Trace* dependency can be used to emphasize that a pair of requirements are related in some way or another. In Figure 9, the requirement for manual deactivation is traced to the one about automatic deactivation. [33]

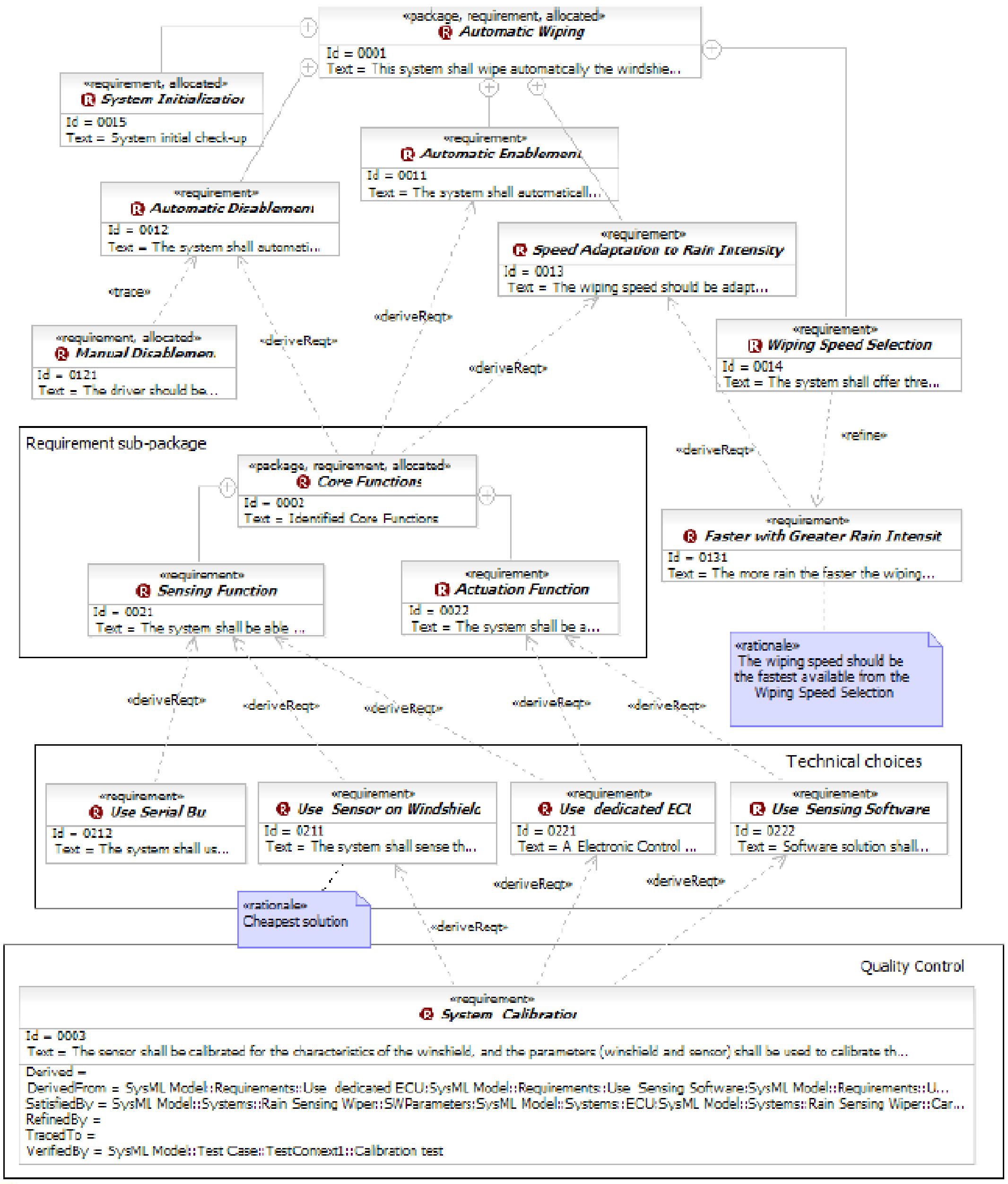

«package, requirement, allocated» Automatic Wiping
Id = 0001
Text = This system shall wipe automatcally the windshie...
«requirement, allocated» System Initializatior
Id = 0015
Text = System initial check-up
«requirement» Automatic Enablement
Id = 0011
Text = The system shall automaticall...
«requirement» Automatic Disablement
Id = 0012
Text = The system shall automati...
«requirement» Speed Adaptation to Rain Intensity
Id = 0013
Text = The wiping speed should be adapt...
«trace»
«requirement, allocated» Manual Disablemen
Id = 0121
Text = The driver should be...
«deriveReqt»
«requirement» Wiping Speed Selection
Id = 0014
Text = The system shall offer thre...
«refine»
Requirement sub-package
«package, requirement, allocated» Core Functions
Id = 0002
Text = Identified Core Functions
«requirement» Faster with Greater Rain Intensit
Id = 0131
Text = The more rain the faster the wiping...
«requirement» Sensing Function
Id = 0021
Text = The system shall be able ...
«requirement» Actuation Function
Id = 0022
Text = The system shall be a...
«rationale» The wiping speed should be the fastest available from the Wiping Speed Selection
Technical choices
«requirement» Use Serial Bu
Id = 0212
Text = The system shall us...
«requirement» Use Sensor on Windshielc
Id = 0211
Text = The system shall sense th...
«requirement» Use dedicated ECL
Id = 0221
Text = A Electronic Control ...
«requirement» Use Sensing Software
Id = 0222
Text = Software solution shall...
«rationale» Cheapest solution
Quality Control
«requirement» System Calibratior
Id = 0003
Text = The sensor shall be calibrated for the characteristics of the winshield, and the parameters (winshield and sensor) shall be used to calibrate th...
Derived =
DerivedFrom = SysML Model::Requirements::Use dedicated ECU:SysML Model::Requirements::Use Sensing Software:SysML Model::Requirements::U...
SatisfiedBy = SysML Model::Systems::Rain Sensing Wiper::SWParameters:SysML Model::Systems::ECU:SysML Model::Systems::Rain Sensing Wiper::Car...
RefinedBy =
TracedTo =
VerifiedBy = SysML Model::Test Case::TestContext1::Calibration test

Figure 9. Requirement Diagram for RSW [33]

After applying derived evaluation criteria [35] on first phase following Evaluation Table 3, is constructed.

Criteria	Support level	Results
Simplicity	H	It become very for different stake holders to express their problems particularly by introduction of requirement diagram.
Explicitness	M	Although SysML provide explicit set of notation for stating problems but still there a need of more Explicitness particularly for modeling complex hierarchy in requirements.
Preciseness	H	Problems particularly constraints can be represented by parametric diagrams.
Completeness	H	As a whole Problem description is facilitated very well by SysML.
Scalability	H	Requirements diagrams supports generalization and dependency relationship in stating problems.
Flexibility	M	Requirements diagrams are flexible enough to support change management issue but there is no special construct is available for change management.

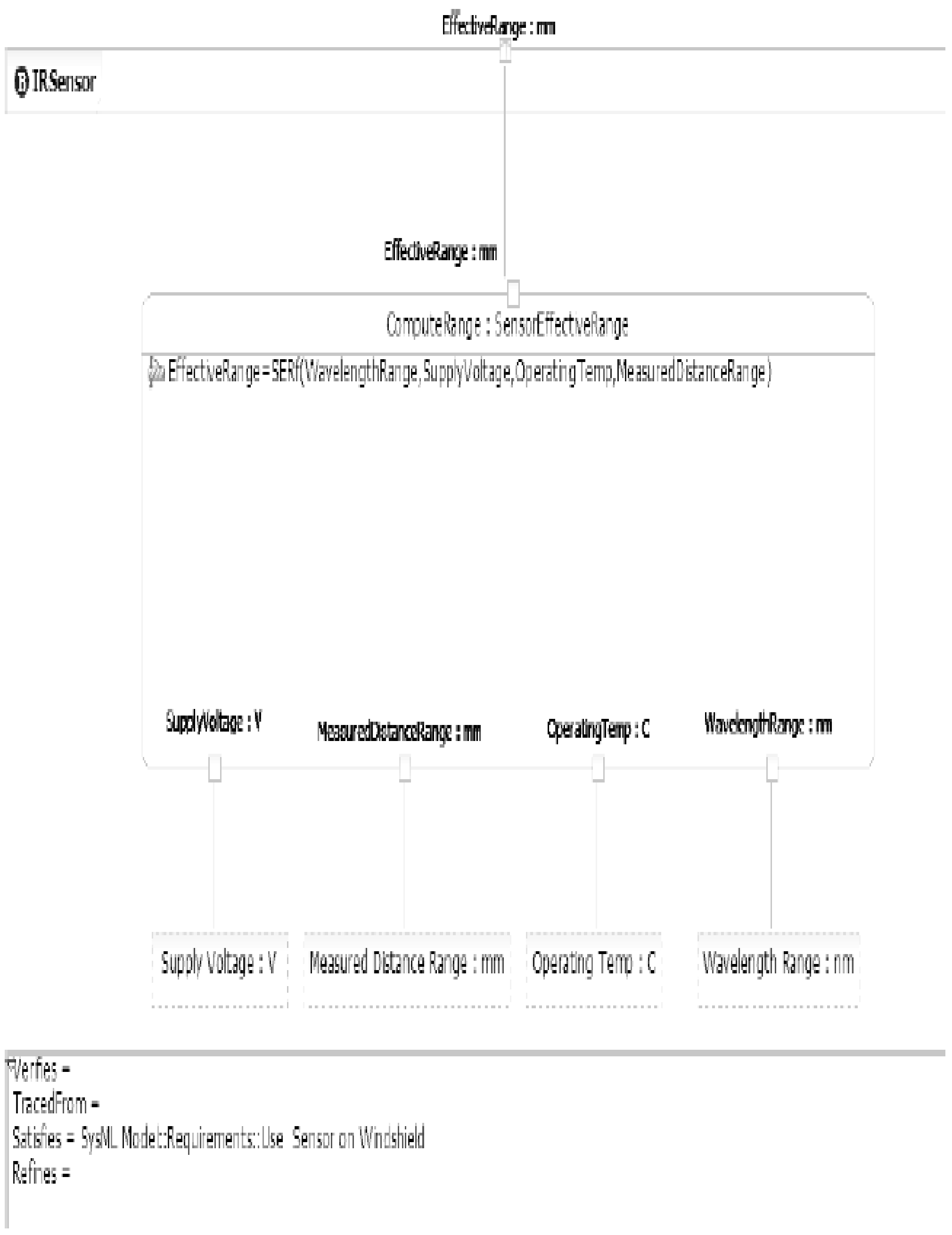
Table 3. Evaluation Table for 'State the Problem'.

6.2 Investigate Alternatives

It is always a good practice to select alternative during construction of design. Generally these parameters are appraised on basis of cost, schedule, performance and different related risks. It should be noted that selection of alternatives is an iterative process i.e. when ever new information comes around, it becomes necessary to analyze alternatives once again. Particularly for large and complex system, consideration of alternatives trims down project risk. Following points are important in considerations of alternatives. [34]

- Evaluation criteria should be prepared in such a way that it can quantify system performance. Traditionally system engineers use a typical system engineering evaluation criteria called figures of merit. SysML parametric diagrams provide a concrete support in establishing Quantitative evaluation criteria for performance measurement. [34]

In general parametric Models are developed in analysis phase to shore up feedback activities, development of performance and reliability models. It is now possible to analyze complex mathematical relationships. Further more system engineers can make comparison between constraints and logical expression, while making decisions for alternatives. [29]


EffectiveRange : mm
IRSensor
EffectiveRange : mm
ComputeRange : SensorEffectiveRange
EffectiveRange=SERf(WavelengthRange,SupplyVoltage,OperatingTemp,MeasuredDistanceRange)
SupplyVoltage : V
MeasuredDistanceRange : mm
OperatingTemp : C
WavelengthRange : nm
Supply Voltage : V
Measured Distance Range : mm
Operating Temp : C
Wavelength Range : nm

Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::Use Sensor on Windshield
Refines =

Figure 10. Parametric models for windshield [33]

Fig.10 shows parametric models for windshield in RSW system, it shows different parameter like range, temperature and wavelength through which engineers in calculating effective range of windshield. This information can be beneficial for make trade-off in selection of windshields. [33]

- It is always necessary to judge how well system reassures its requirements. Actually requirements based evaluation criteria is established to measure effectiveness of system. In general these criterions are established to quantify system requirements [3] [4]. SysML requirement diagram provide a good support for making trade-off, setting priorities among requirements. So, it becomes a vital consideration in establishing evaluation criterion in present scenario. [29]

- One of important consideration in system engineering process is technical performance measures (TPM's). These measures are conducted to estimate the probability of fulfilling requirements. It should be noted that TPM is not conducted for all requirements, but it is applied on certain specific requirements particularly if certain threats are associated with these requirements. [34]

After applying derived evaluation criteria [35] on second phase following Evaluation Table 4, is constructed.

Criteria	Support level	Remarks
Simplicity	L	Alternative investigation is mainly done by analyzing different constraints through parametric diagram but there diagram are very mush technical which may not be understood by customers
Explicitness	L	For selecting alternative market scenario, product positions can be considered explicitly in SysML.
Preciseness	H	Mathematical expressions can be easily modeled for conducting different types of trade-off.
Completeness	M	Although Language provides good support for investigating alternatives but business scenario can not be easily plotted in SysML.
Scalability	H	The language support verity of relationships in alternative selection.
Flexibility	M	Different options regarding alternatives can be easily considered in SysML but situation become complex as low level details are added.

Table 4. Evaluation Table for 'Investigate Alternative'.

6.3 Model the System

INCOSE define the System engineering as *"Systems engineering integrates all the disciplines and specialty groups into a 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."* [42]

From the above mentioned definition, it becomes clear that system engineering not only focus on product development but it also pay good attention to process through which products are developed. So modeling activities in context of system engineering split up into two parts i.e. process modeling and product modeling. Where product models explicate the system under consideration and these models are also helpful while making trade-off and in handling different associated risks. Process models not only disclose different constraints and disjointed activities but they also provide a good support for reducing cost and in exposing duplication of effort. [34]

Traditionally system engineers develop several types of system models like functional flow diagrams, object-oriented models and computer simulations. [34]

Now system engineers can utilize SysML block diagram for partitioning system into different sub-systems. Actually blocks are basic building unit of system in SysML that can describe structural and behavioral feature for epitomizing state of system. It should be noted that block provide general-purpose competence for Modeling verity of elements like hardware, software and bioware or wetware. Actually bioware are human aspect or other biological organism that can be element of system. Important property of SysML blocks is that they can be used through out life cycle and work very well for different kind of system. Block Definition Diagram (BDD) captures feature of block in terms of its properties, operations and relationships like generalizations, associations and dependencies. [1]

Block definition diagram for RSW can be developed in following manner, where RSW is decomposed into three blocks i.e. external sub-system, rain sensing wiper and composite sub-system. As block can contain arbitrary no sub-blocks for this reason "composite sub-system" comprised of sensor attachment, IRsensor, software configuration, rain sensing wiper software and Electrical Configuration Unit (ECU).[33]

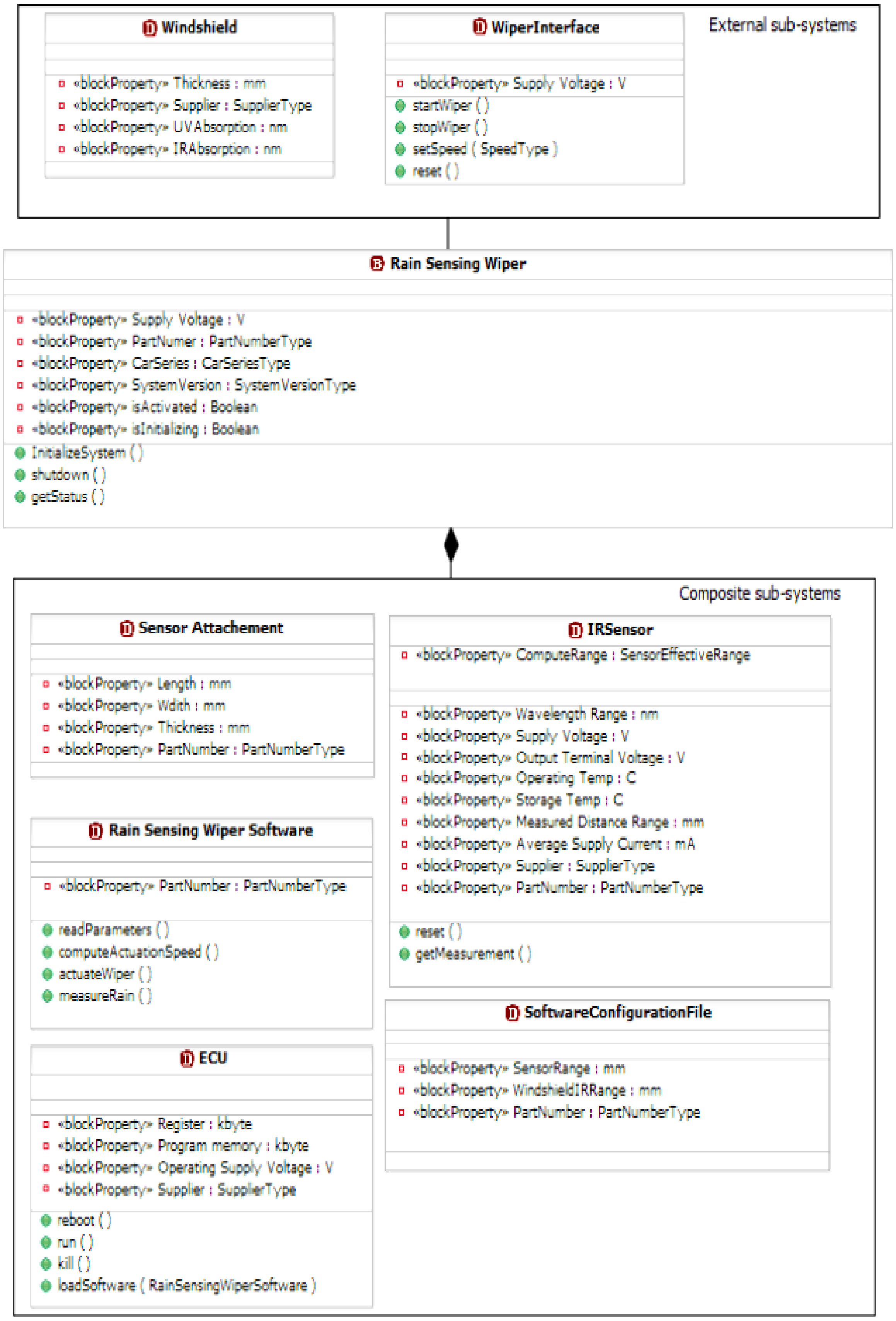

Windshield
«blockProperty» Thickness : mm
«blockProperty» Supplier : SupplierType
«blockProperty» UVAbsorption : nm
«blockProperty» IRAbsorption : nm
WiperInterface
«blockProperty» Supply Voltage : V
startWiper ( )
stopWiper ( )
setSpeed (SpeedType)
reset ( )
External sub-systems
Rain Sensing Wiper
«blockProperty» Supply Voltage : V
«blockProperty» PartNumer : PartNumberType
«blockProperty» CarSeries : CarSeriesType
«blockProperty» SystemVersion : SystemVersionType
«blockProperty» isActivated : Boolean
«blockProperty» isInitializing : Boolean
InitializeSystem ( )
shutdown ( )
getStatus ( )
Composite sub-systems
Sensor Attachement
«blockProperty» Length : mm
«blockProperty» Width : mm
«blockProperty» Thickness : mm
«blockProperty» PartNumber : PartNumberType
Rain Sensing Wiper Software
«blockProperty» PartNumber : PartNumberType
readParameters ( )
computeActuationSpeed ( )
actuateWiper ( )
measureRain ( )
ECU
«blockProperty» Register : kbyte
«blockProperty» Program memory : kbyte
«blockProperty» Operating Supply Voltage : V
«blockProperty» Supplier : SupplierType
reboot ( )
run ( )
kill ( )
loadSoftware (RainSensingWiperSoftware)
IRSensor
«blockProperty» ComputeRange : SensorEffectiveRange
«blockProperty» Wavelength Range : nm
«blockProperty» Supply Voltage : V
«blockProperty» Output Terminal Voltage : V
«blockProperty» Operating Temp : C
«blockProperty» Storage Temp : C
«blockProperty» Measured Distance Range : mm
«blockProperty» Average Supply Current : mA
«blockProperty» Supplier : SupplierType
«blockProperty» PartNumber : PartNumberType
reset ( )
getMeasurement ( )
SoftwareConfigurationFile
«blockProperty» SensorRange : mm
«blockProperty» WindshieldIRRange : mm
«blockProperty» PartNumber : PartNumberType

Figure 11. RSW Block Definition Diagram [33]

Functional decomposition is another important activity in modeling. System engineers do functional decomposition for mapping functions to physical components and to system requirements. Engineers also confirm that all necessary tasks are taken in to account and irreverent details have been cut down. [34]

SysML block definition diagram, Internal Block Diagram (IBD) and requirement diagram can play significant role in functional decomposition. System can be decomposed explicitly in term of block with help of block definition diagram; further more engineers can model internal structure of block by plotting internal block diagrams. For example Internal Block diagram of for block “rain sensing wiper power station” can be developed as follows. [33]

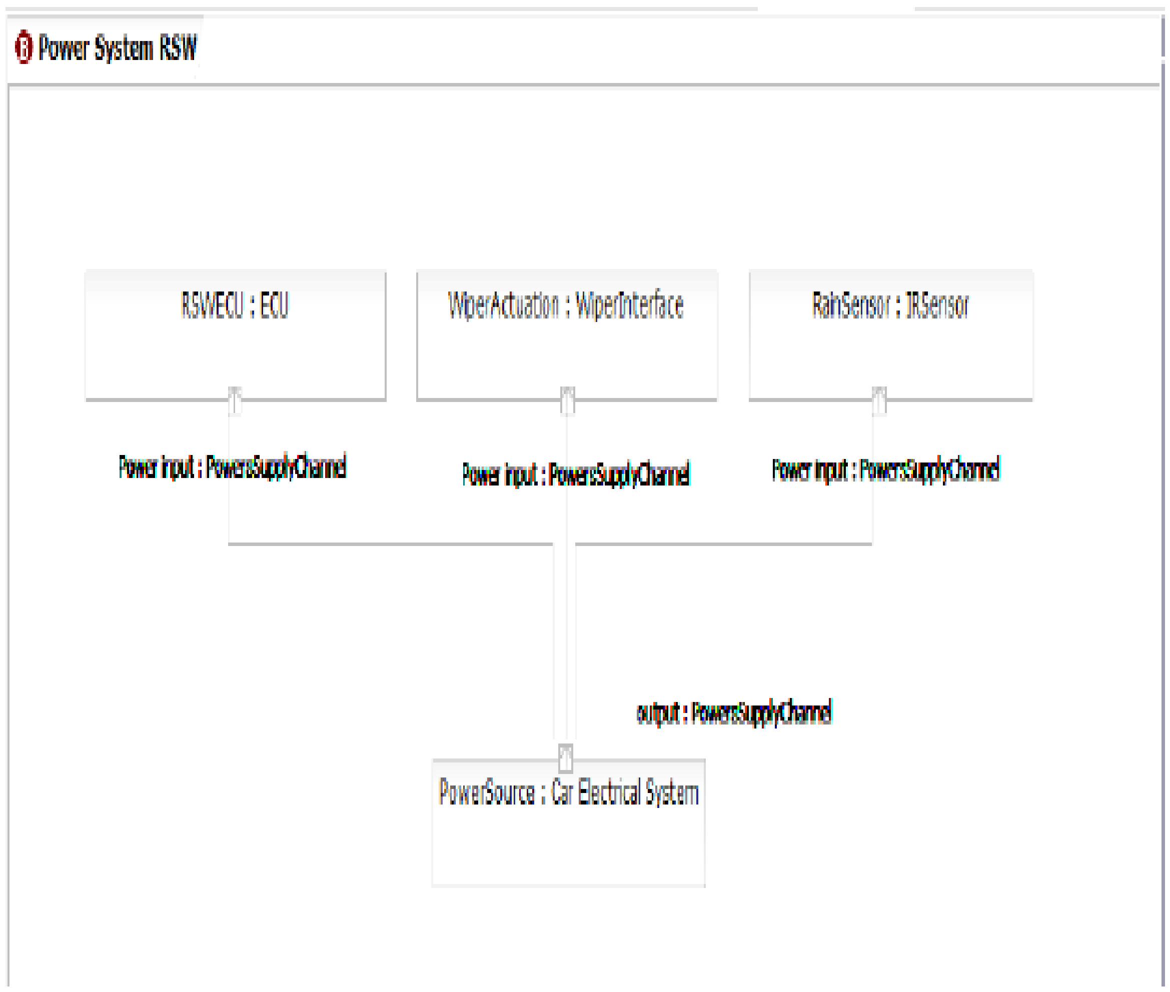
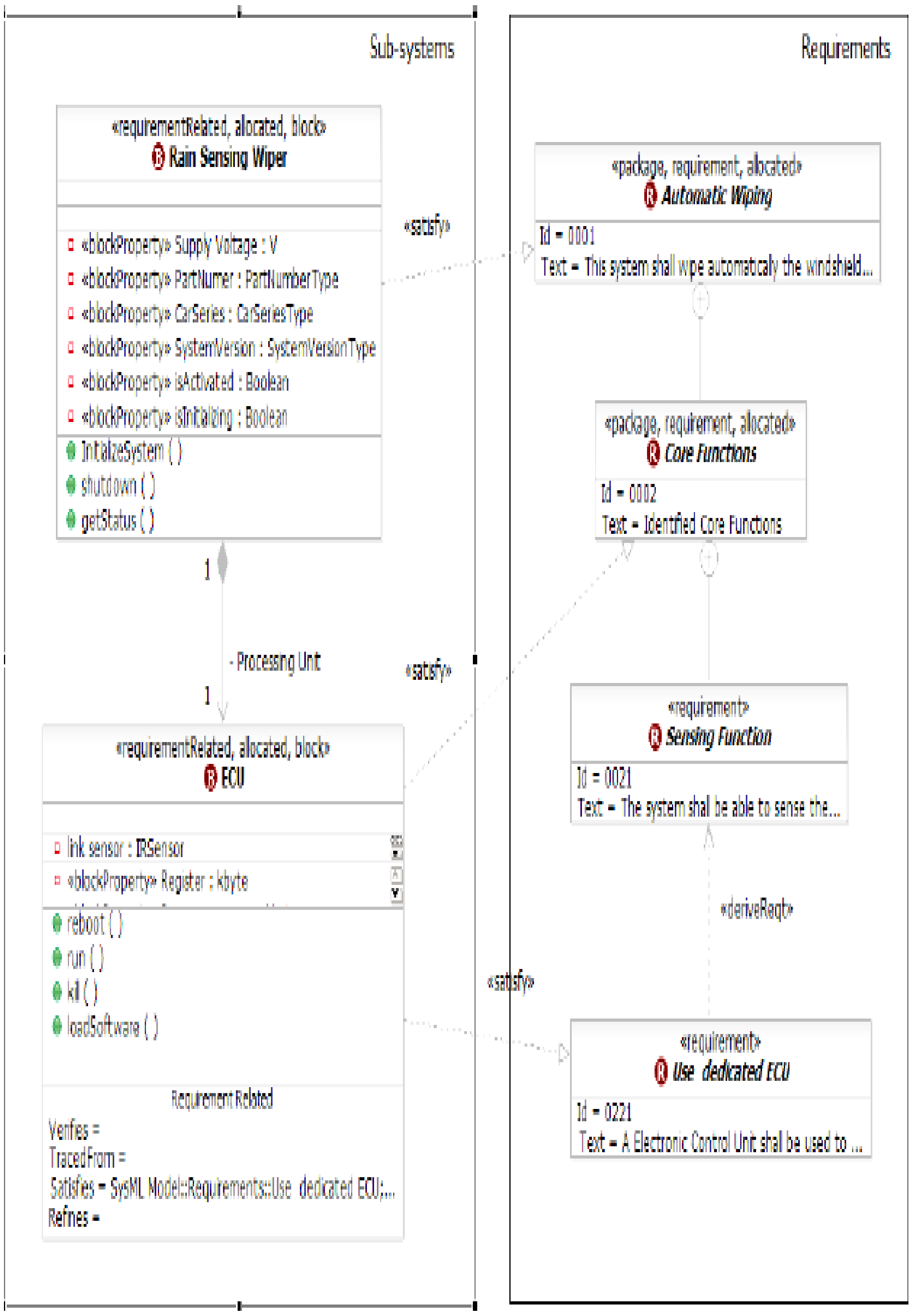

Power System RSW
RSWECU : ECU
WiperActuation : WiperInterface
RainSensor : IRSensor
Power input : PowersSupplyChannel
Power input : PowersSupplyChannel
Power input : PowersSupplyChannel
output : PowersSupplyChannel
PowerSource : Car Electrical System

Figure 12. Internal Block Diagram (IBD) RSW[33]

It is obvious from Fig 12 that now it is possible for engineers to get inside system structure and can solve complex structural issues more efficiently. Further more now it become a straightforward issue to map requirements to corresponding functions with help of allocations. Allocation help designer in establishing mapping between structural elements and requirements. For RSW allocations between requirements and structural elements can be represented by following fig.13.

where on left hand side shows products elements and on right side requirements are present. <<satisfy>> dependency is established to carry out allocations. Diagram further shows that “rain sensing wiper” block is allocated to requirement “automatic wiping”. In fact <<satisfy>> dependency means that design elements should satisfy their corresponding requirements. [33]



Sub-systems
Requirements
«requirementRelated, allocated, block»
Rain Sensing Wiper
«blockProperty» Supply Voltage : V
«blockProperty» PartNumer : PartNumberType
«blockProperty» CarSeries : CarSeriesType
«blockProperty» SystemVersion : SystemVersionType
«blockProperty» isActivated : Boolean
«blockProperty» isInitializing : Boolean
InitalizeSystem ( )
shutdown ( )
getStatus ( )
1
- Processing Unit
1
«satisfy»
«satisfy»
«satisfy»
«requirementRelated, allocated, block»
ECU
link sensor : IRSensor
«blockProperty» Register : kbyte
reboot ( )
run ( )
kill ( )
loadSoftware ( )
Requirement Related
Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::Use dedicated ECU;...
Refines =
«package, requirement, allocated»
Automatic Wiping
Id = 0001
Text = This system shall wipe automatically the windshield...
«package, requirement, allocated»
Core Functions
Id = 0002
Text = Identified Core Functions
«requirement»
Sensing Function
Id = 0021
Text = The system shall be able to sense the...
«deriveReqt»
«requirement»
Use dedicated ECU
Id = 0221
Text = A Electronic Control Unit shall be used to ...

Figure 13. Allocations [33]

SysML can also look up value engineering [34] activities, because when engineers analyze exiting system in order to improve its performance they always perform functional decomposition. They suggest changes how design can be changed in order to achieve performance goals.

It should be noted that Blocks in SysML enjoy reusability characteristic, which make it possible for designers whether to make use of exiting sub-system or buy some COTS components form market. [34]

SysML retains UML sequence diagrams through which it is possible to collect sequence of events through which system will pass in particular scenario. It is always a continent practices for system engineers to transform sequence diagram in to system design. [1]

After applying derived evaluation criteria [35] on third phase following Evaluation Table 5, is constructed.

Criteria	Support level	Remarks
Simplicity	M	Technical details can not be easily understood by customers.
Explicitness	H	SysML 'Block' can represent verity of elements like hardware, software, people and data. In short Blocks provide more explicitness to system model during the SIMILAR [ ] phase "Model the System".
Preciseness	H	Parametric diagram provides good support mathematical relationships.
Completeness	L	Business interest can not be modeled easily.
Scalability	H	Good support for dependency and generalization.
Flexibility	H	Good support for change management in this stage.

Table 5. Evaluation Table for 'Model the System'

6.4 Integrations

During integration system engineers develop relationships among all components of system so that these components can work as system. It is worth while to mention here that major objective of integration is to ensure that system will produce desired result and sub-systems should not conflict with each other.

Actually during integration interfaces establish among different sub-system and between main system and external environment. According to system engineering point of view sub-systems should made as self consistent as possible i.e. they only send finished information to each other and depend on each other to minimum extent.[34]

SysML introduced several concepts that can facilitate engineers during integration like ports, value binding for mathematical equations, allocation of function. Common objectives of all these mechanism are to ensure completeness and consistency in resultant model. [26]

One key concept given by SysML for integration is port; actually port is a point of interaction thorough which block can make interaction to other blocks or to its environment. Actually ports are such part of block that that grants access to internal structures from outer environment it should be noted that port makes block reusable by defining its interfaces. Further more port is part of block and a block can have several ports. [1]

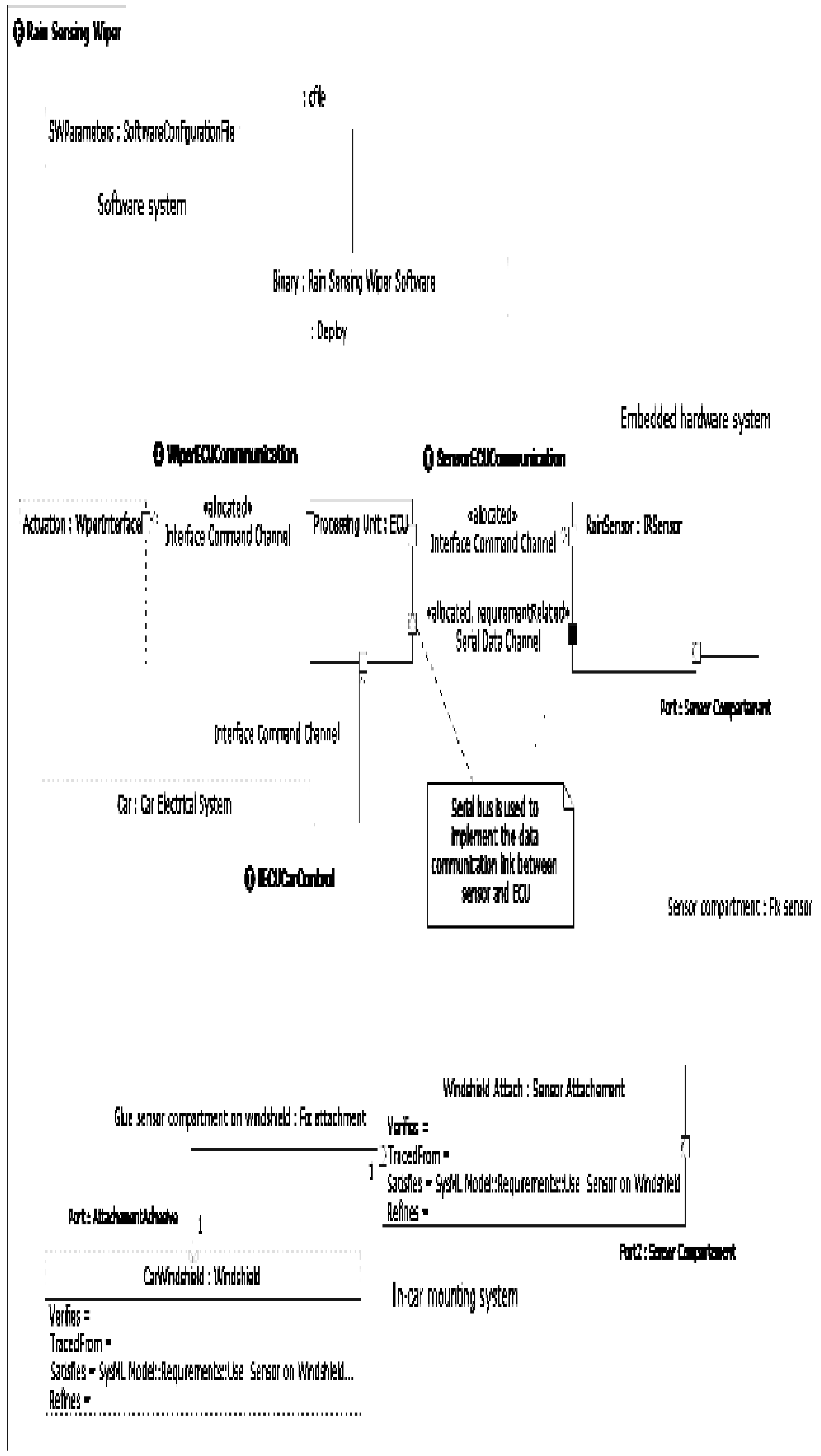

Rain Sensing Wiper
: cfile
SWParameters : SoftwareConfigurationFile
Software system
Binary : Rain Sensing Wiper Software
: Deploy
Embedded hardware system
WiperECUCommunication
SensorECUCommunication
«allocated»
Interface Command Channel
Actuation : WiperInterface
Processing Unit : ECU
«allocated»
Interface Command Channel
RainSensor : IRSensor
«allocated, requirementRelated»
Serial Data Channel
Port : Sensor Compartment
Interface Command Channel
Car : Car Electrical System
Serial bus is used to implement the data communication link between sensor and ECU
ECUCarControl
Sensor compartment : Fix sensor
Windshield Attach : Sensor Attachement
Glue sensor compartment on windshield : Fix attachment
Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::Use Sensor on Windshield
Refines =
Port : AttachementAdhesive
Port2 : Sensor Compartment
CarWindshield : Windshield
In-car mounting system
Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::Use Sensor on Windshield...
Refines =

Figure 14.Ports and Interfaces [33]

Fig.14 shows different hardware and software part of RSW. For establishing communication among different part RSW a set of ports and interface is defined. Usually a complex model comprise of hundreds and thousands ports and interface. [14]. For instance processing unit develops contact with actuation interface with help of *WiperECUCommunication* interface. [33]

Form the INCOSE definition of system engineering [11], it become obvious that one of major objective of system engineering is to deals with growing complexities due to systems expansion. SysML port can be used to developed complex hierarchical structures which are not possible in UML, so integration complexities can be addressed in a disciplined manner with the help of ports as shown in fig.14. [29, 33]

During integration now system engineers can enjoy another latest concept given by SysML i.e. Allocation. Through this relationship, now system engineers can allocate group of model elements to one another, for example allocation of behavior to structure or logical components to physical. [43]

Now it is possible for system engineers to establish controlled cross-association of elements across different structures and hierarchies. It should be noted that allocation can be applied to abstract specifications. For this reason allocation is equally applicable to all types of elements i.e. whether they are hardware, software or bioware. Further more allocation can be used to originate ideas regarding system specifications. [43]

After applying derived evaluation criteria [35] on fourth phase following evaluation table 6, is constructed.

Criteria	Support level	Remarks
Simplicity	M	Even though SysML is strongly based on UML, due to some new constructs stakeholders may face some problems.
Explicitness	H	SysML provide concept of “allocations” which makes integrations quite explicit.
Preciseness	H	Different constraints can be easily integrated with help of parametric diagrams.
Completeness	M	Although allocation may be applied to all kinds of elements but complexities may occur if system has deep levels in Work Breakdown Structures (WBS).
Scalability	H	Enhancement in UML activity diagram and introduction of “allocations” make it possible to address complex integration problems.
Flexibility	M	Change during integration is not an easy task. How SysML “Block” and “allocation’ can support change management issues during integration.

Table 6. Evaluation Table for ‘Integration’

6.5 Launch the System

Actually launching implies executing the system for gaining desired output. It actually the system engineering of deployment related activities. To avoid any hazard system engineers designed several alternative for this stage. Components which remains unable to get certified level of satisfaction from engineers and customers will be replaced. [9]

In short configuration management process comes in to play. In general configuration management deals with changes in requirements and make sure that implementation is controlled and properly recorded. Actually system engineering process is iterative and it possible to make changes at any specific point but it is necessary to get consent from all stake holders. [34]

SysML requirement diagrams and block diagrams can provide a good support to system engineering activities at this stage. It is obvious that a requirement tracking is an important activity in configuration management. <<Trace >> relationship in requirement diagrams provides a good support in this regard. Actually <<trace>> is very flexible relationship and can be established between requirement and any other model element. Now it is possible for system engineers to develop traceability matrix by requirement tracing capability of SysML requirements diagrams. [1]

An example of <<trace>> dependency can be observed in case study of RSW. Actually RSW contain a quality requirement called "system calibrations" stating that system should be evaluated according to certain specific parameters like voltage, sensor range, and windshield range. So, it necessary that sensor and windshield should be very flexible i.e. easy to change but whenever they change it should be done according to parameters mentioned above.[33]

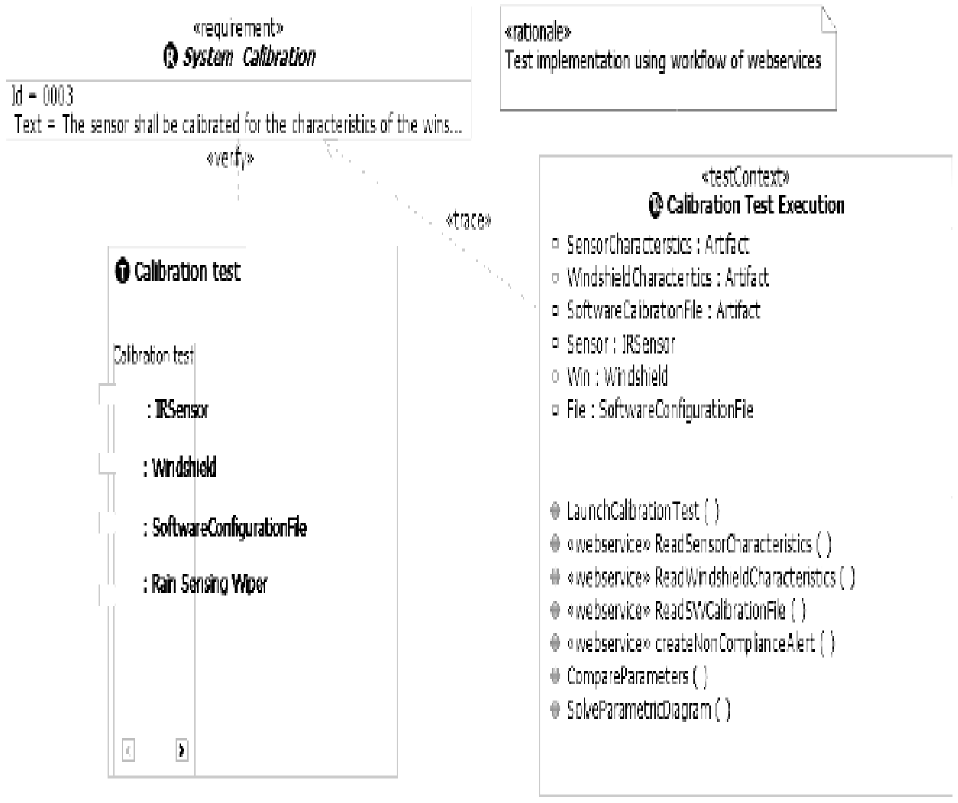

«requirement»
System Calibration
Id = 0003
Text = The sensor shall be calibrated for the characteristics of the wins...
«rationale»
Test implementation using workflow of webservices
«verify»
«trace»
Calibration test
Calibration test
: IRSensor
: Windshield
: SoftwareConfigurationFile
: Rain Sensing Wiper
«testContext»
Calibration Test Execution
SensorCharacteristics : Artifact
WindshieldCharactertics : Artifact
SoftwareCalibrationFile : Artifact
Sensor : IRSensor
Win : Windshield
File : SoftwareConfigurationFile
LaunchCalibrationTest ( )
«webservice» ReadSensorCharacteristics ( )
«webservice» ReadWindshieldCharacteristics ( )
«webservice» ReadSWCalibrationFile ( )
«webservice» createNonComplianceAlert ( )
CompareParameters ( )
SolveParametricDiagram ( )

Figure 15. Test case for RSW [33]

system calibration requirement can be verified by test case as shown if Fig.15 test case have two part; first it will test presence of different components like software configuration file, sensor, windshield. Secondly, test case will also calculate ranges for sensor and wind shield, and evaluate whether they are compatible or not.[33]

Now it is obvious that SysML requirement diagram can provide a good support for verification and validation. It is possible through requirement diagram to associate requirements to appropriate test cases. Requirement diagram may also valuable in conduction sensitivity analysis. In this type of analysis each requirement is verified with respect to its effects on out put of system [34].

During system launching system engineers can enjoy the benefits of reusability attribute of blocks and can make appropriate adjustments in system structure. General-purpose potential to model system in terms of components. Further more they can be used through out system development life cycle It should be noted that with help of internal block diagram engineers can capture inner details of block structure which help them in make trade-off and reusability decisions. [1]

After applying derived evaluation criteria [35] on fifth phase following Evaluation Table 7, is constructed

Criteria	Support level	Remarks
Simplicity	H	All involving requirements is represented by requirement diagram clearly, stakeholders can capture good idea of requirements against which system is being launched.
Explicitness	H	V & V related issues play dominant role during launching of system. These activities can be represented explicitly by SysML through requirement diagram.
Preciseness	H	Mathematical representation of constraints makes V and V result more precise and reliable.
Completeness	M	The language support different interesting during launching like testing, change management but remains to extend much support for market analysis and product strategies.
Scalability	H	Yes, SysML can support the Launching of scalable products as requirements are described in clear cut manner. Further more Blocks can help in breaking system into several systems.
Flexibility	H	As system comprised of several independent components which are loosely coupled so changes can be made in disciplined manner.

Table7. Evaluation Table for 'Launch the System'

6.6 Assess Performance.

Development of performance assessment criterion is one important factor in system engineering process. According to Bahill and Dean *"If you cannot measure it, you cannot control it. If you cannot control it, you cannot improve it"*. [34]

Conventionally in system engineering employ several performance evaluation methods like technical performance measures, figures of merit and different types of matrices. System engineers use technical performance to diminish different risks which come on to surface during development. For making trade-off, traditionally system engineers rely on figures of merit. However different kinds of metrics also play an important role in performance assessment. Some of important metrics are concerning with productivity, number of problem and customer satisfaction levels. [9]

SysML provide excellent support in establishing performance criterion for a certain system particularly requirement diagram and parametric diagram can be very useful. Parametric diagram supports the development of evaluation criterion. Through these diagrams engineers can model properties and relationships in term of complex mathematical or logical relationships. In general parametric models are established during analysis phase. These models provide a good support in creating different engineering models like performance, reliability and cost models. [29, 1,36]

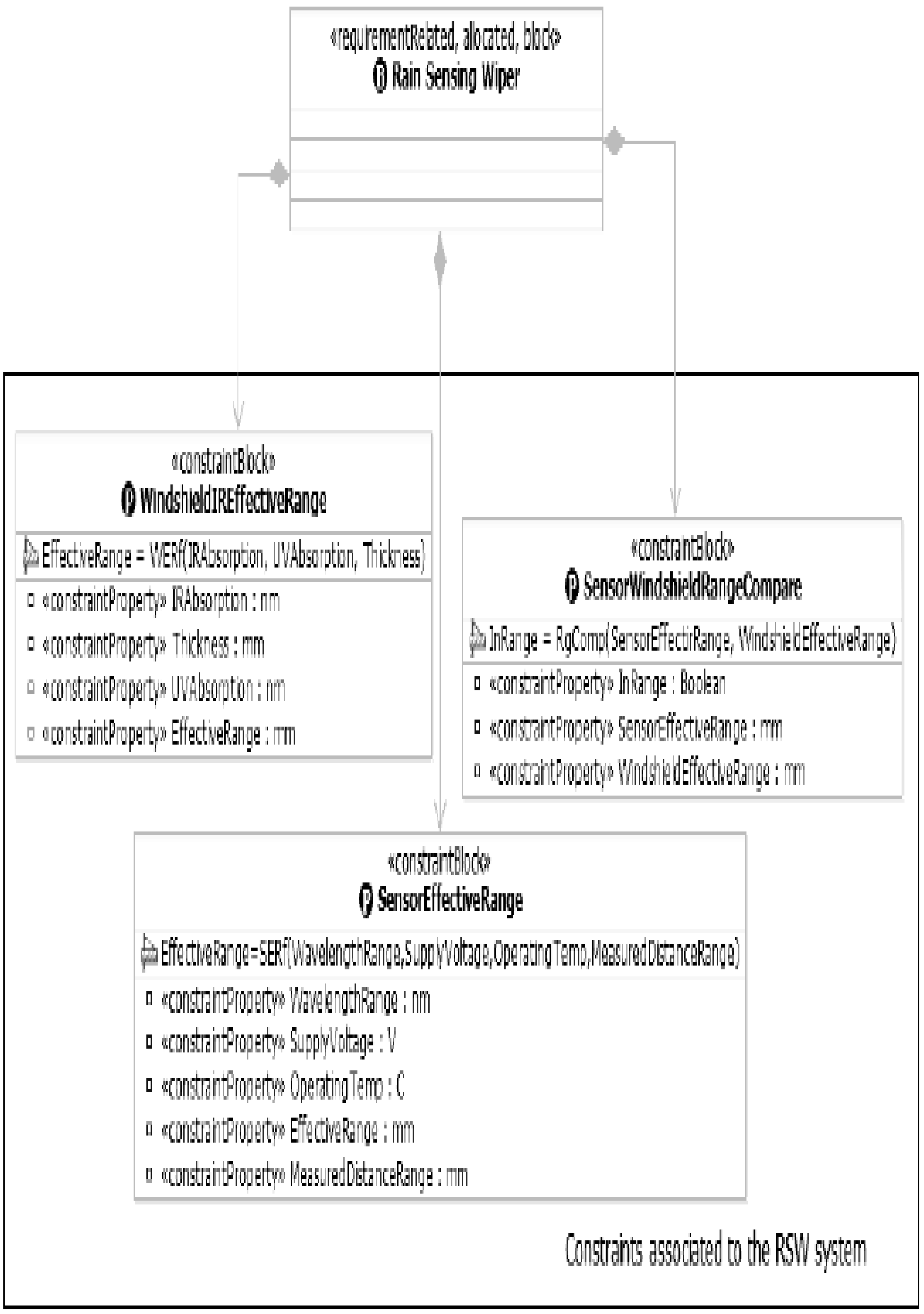

«requrementRelated, allocated, block»
Rain Sensing Wiper
«constraintBlock»
WindshieldIREffectiveRange
EffectiveRange = WERf(IRAbsorption, UVAbsorption, Thickness)
«constraintProperty» IRAbsorption : nm
«constraintProperty» Thickness : mm
«constraintProperty» UVAbsorption : nm
«constraintProperty» EffectiveRange : mm
«constraintBlock»
SensorWindshieldRangeCompare
InRange = RgComp(SensorEffectiRange, WindshieldEffectiveRange)
«constraintProperty» InRange : Boolean
«constraintProperty» SensorEffectiveRange : mm
«constraintProperty» WindshieldEffectiveRange : mm
«constraintBlock»
SensorEffectiveRange
EffectiveRange=SERf(WavelengthRange,SupplyVoltage,OperatingTemp,MeasuredDistanceRange)
«constraintProperty» WavelengthRange : nm
«constraintProperty» SupplyVoltage : V
«constraintProperty» OperatingTemp : C
«constraintProperty» EffectiveRange : mm
«constraintProperty» MeasuredDistanceRange : mm
Constraints associated to the RSW system

Figure 16.Constraints Blocks for RSW [33]

Actually Parametric model is a heavily relying on block definition diagram. Fundamental unit of parametric model is constraints block. While establishing parametric model, first different constraints block are realized, then explore details of each blocks in terms of parametric diagrams which is specialized version of Internal Block Diagram (IBD). The constraints blocks allow engineers not only to express constraints but also to develop analytical relationship between constraints. Where constraints are properties in blocks, named as *ConstraintProperty* and are typed by *ConstaintBlocks.* In RSW requirement of *system calibration* is verified by a set of constraints. Fig 10 clearly shows three constraints Blocks; (i) *SensorEffectiveRange* which computes an operational range for sensor (ii) *WinshieldIREffectiveRange* which is responsible to calculate range for infrared sensor (iii) *SensorWindshieldRangeCompare* above values are compared by this constraint.

Parametric diagram is developed by manipulating inside details of constrains blocks. Actually these diagrams make use of constraint blocks as constraint properties. Parametric diagram for RSW is shown in fig17. [33]

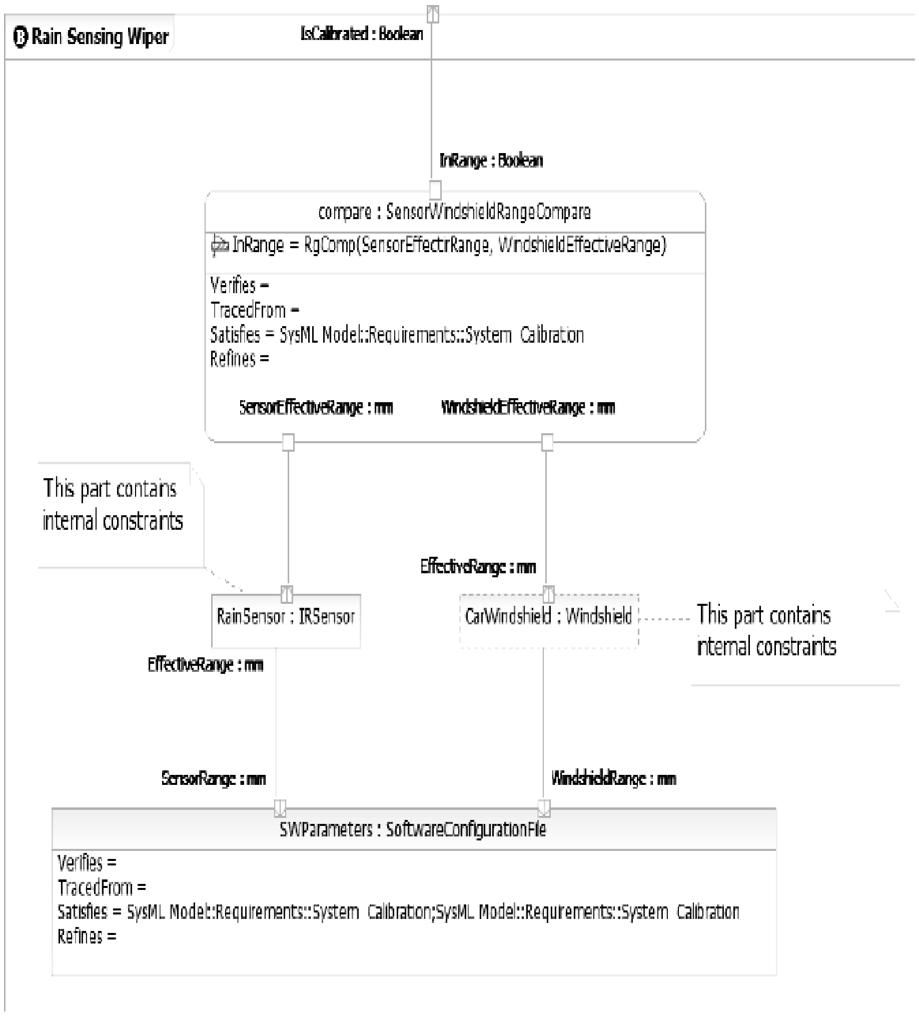

Rain Sensing Wiper
IsCalibrated : Boolean
InRange : Boolean
compare : SensorWindshieldRangeCompare
InRange = RgComp(SensorEffectrRange, WindshieldEffectiveRange)
Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::System Calibration
Refines =
SensorEffectiveRange : mm
WindshieldEffectiveRange : mm
This part contains internal constraints
EffectiveRange : mm
RainSensor : IRSensor
CarWindshield : Windshield
This part contains internal constraints
EffectiveRange : mm
SensorRange : mm
WindshieldRange : mm
SWParameters : SoftwareConfigurationFile
Verifies =
TracedFrom =
Satisfies = SysML Model::Requirements::System Calibration;SysML Model::Requirements::System Calibration
Refines =

Figure 17. Parametric diagram for RSW [33]

It is already mentioned that parametric model helps a lot in performance assessments. For example in Fig.17, property "compare" operational ranges of sensor and windshield and give clear idea regarding performance of RSW. [33]

It is always desirable that system engineering should expose those tests which will be conducted to demonstrate that system is complying with its requirements. Through SysML requirement diagram it is very much possible to define test cases with each requirement. Further more, requirements diagram provides strong support in conduction of verification and validation activities which make it possible for Systems Engineers to prove that the final system fulfills each system requirement.

It becomes very convenient for system engineers to conduct trade-off analysis with help of parametric models. The constraint block which is fundamental unit of this model is used to compare different alternative way out. [29, 1]

After applying derived evaluation criteria [35] on sixth phase following Evaluation Table 8, is constructed

Criteria	Support level	Remarks
Simplicity	H	All stakeholders can have definite idea regarding assessment with help of parametric diagrams.
Explicitness	H	Parametric diagram helps engineer in conducting performance measurement more explicitly. For this reason assessment results become accurate.
Preciseness	H	Constraints can be easily translated into mathematical expressions.
Completeness	H	SysML can support verity of interest and activities during performance assessment.
Scalability	H	Complex constraints can be easily dealt for large scale system development.
Flexibility	H	Different values for performance parameters can be uses e.g. different values for response time can be experimented.

Table8. Evaluation Table for 'Assess Performance'

6.7 Re-evaluation

As it is described earlier that system engineering process is an iterative process which comprise of many parallel loops. It should be noted that ultimate aim of system engineering is to achieve enhanced levels of performance by satisfying system requirements. Through re-evaluation system engineers continuously examine output of system. If there is any flaw in system productivity engineers will make decisions about modification of system inputs. Actually engineers establish feedback loops for re-evaluation and through these loops they will keep close eye on system performance. [9]

Conventionally system engineers relying on several techniques like, Deming's quality improvement concepts, Quality Function Deployment (QFD) and Total Quality Management (TQM) [3]. Now they can also utilize SysML for establishing continuous quality improvement frame work because Language offer flexible way to model different aspect of system. For instance it is possible in SysML to model text-based requirement and their corresponding relationship in flexible manner. << *Derive* >> relationship makes it possible to explore new requirements in analysis phase. For example if new requirements for RSW come around then it seems to very flexible that engineers can easily accommodate these requirements into exiting requirement model. [33]

It is never be an easy task to execute a continuous quality improvement program if system structure is complex and not understandable. Particularly when several disciplines are involved in system structure. SysML provide Block definition diagram which can represent system structure in a very straightforward way. These diagrams solve complexities issues by providing different relationship like association and generalization. Actually SysML block is very flexible construct which can represent hardware, software and bioware. So, it possible to model different kind sub-system and components in uniform manner. Further more blocks are provided with ports which act as interface to blocks. [33]

After applying derived evaluation criteria [35] on seventh phase following evaluation table 9 is constructed

Criteria	Support level	Remarks
Simplicity	H	As long as system is in practical use, SysML can provide effective way to understand different aspect of system.
Explicitness	M	Requirement diagram and Block diagram provides very straightforward notations for re-evaluations.
Preciseness	H	Parametric diagram generates precise expression for continuous improvement of quality.
Completeness	M	Improvement in product strategies can be addressed directly during re-evaluations.
Scalability	H	Parametric diagram can solve performance measurement problem of large complex systems.
Flexibility	H	Block diagram can help engineers in accommodating changes which come on surface during re-evaluations.

Table9. Evaluation Table for 'Re-evaluation'

7 DISCUSSION AND FUTURE WORK

7.1 Discussion

Foremost objective of this study is to analyze effectiveness of SysML for system engineering applications. For this purpose study is divided into different steps like in first step differences between software engineering and system engineering are highlighted and also try to find motivations for adoption of system engineering. In next step architecture of SysML is explored and then formulates some evaluation criteria through which SysML can be analyzed.

Actually competitive pressure and increasing quality consciousness of customers forces the engineers to think beyond boundaries of software. Furthermore term like systems-of-systems (SoS) clearly indicates growing complexities in system structures. In such circumstances, engineers should make an effort to shift their focus from software to system.
Conventionally system engineering is considered to be documents centric process, involving broad range of documents which cover nearly all activities of system engineering process. Some of significant documents are specification, design documents, verifications plans and organizational procedures. Over the years system engineers have been relaying on different modeling techniques like block diagrams, behaviour diagrams and Enhanced Functional Flow Block Diagrams (EFFBD). [35, 32]

One of the modern definitions of system engineering has been given by INCOSE [11], actually this definition has a strong emphasis on current industrial situation where system are becoming so much complex and large that they can not be realized by traditional modeling techniques. During last decade system engineers were reliant on UML for modeling purposes because UML is a general-purpose modeling language and is applicable to verity of domains. Major problem in UML is that it is "software centric" i.e. More appropriate for modeling software intensive systems. It should be noted that complex system may include non-software elements like hardware, people, information and procedures and it is true that UML can not address there issues properly due to its software bias. System engineers have always been in search of certain domain specific language which can not only address software issues but also have a capability of dealing with problem relating to non-software aspect of system. [17]

SysML is jointly developed by OMG [2] and INCOSE [11] as domain specific language for system engineering. Although SysML is strongly based on UML but it reduced size of UML as it contains nine diagram where as UML has thirteen diagrams. Further more it also reduced software bias of UML. SysML also extended UML semantics by including Requirement and parametric models. These two new added diagrams are actually providing a solid support in carrying out requirement engineering and performance analysis, two very important system engineering activities. Support for automated V & V and gap analysis another feature of SysML, for this language provides concept of allocations. It seems to be valuable to mention that SysML does not recommend any development process for its application but it is up to designer to select appropriate process to apply language. [17]

In spite of above mentioned encouraging factors SysML constructs support IEEE-Std-1471-2000 (IEEE Recommended Practice for Architectural Description of Software Intensive Systems) which in model management in case of complex problem. More over SysML base Model are always flexible and easy to change, so it become easy for engineers to conduct configuration management activities. [44]
ISO AP233 is compatible with SysML specification. It should be noted that ISO AP233 is data exchange protocol for transferring system engineering data. Actually real benefit of compatibility is to ensure that SysML models can be switched over from one place to another via the AP233 data exchange protocol. AP233 also help by making SysML tool friendly i.e.

data generated by certain tool which is AP233 complainant can be viewed in SysML tool. [43][41]

As the major intention of this study is to find out degree of effectiveness of SysML for system engineering applications. For this purpose INCOSE recommended system engineering process SIMILAR [9, 34] is focused. The seven phases of SIMILAR represent different kinds of activities. For exploring extent to which the SysML can support different activities in context of SIMILAR [9, 34]. An evaluation criterion has been derived from Friedenthal and Burkhart [35] frame work for assessing capabilities of modeling languages. To support research result and for making them more reliable and applicable, a case study of Rain Sensing System (RSW) [33] is considered.

Foremost consideration of this study is to investigate why SysML is better than UML for system engineering applications. Actually systems in current industrial scenario contain several disciplines in their structure like hardware, software, people and procedures. To address all these area some modification have been made in exiting structure of UML. Through sub-question 3 these modification have been explored. Another important consideration is to explore how these modifications have been incorporated in SysML architecture. Probe in this matter gave explanations to sub-question 2. Software-centric attitude is one of significant objection on UML i.e. it works very well for software application but show weakness to address all aspect of system like hardware, people, procedures. To apply SysML on system it is necessary to make clear distinction between software engineering and system engineering. The differentiation between software engineering and system engineering has been made by considering sub-question 1. By taking in to account information through these sub-question, main research question is answered by deriving evaluation criteria from exiting knowledge [35] and apply it on SIMILAR [9, 34] in light of case example [33].

According to newly derived criteria, SysML is evaluated against certain points like explicitness, preciseness, completeness, simplicity, scalability and flexibility in context of SIMILAR [9, 34]. In fact system engineering is not only responsible for creating such products which can achieve maximum level of customer satisfaction but also it guarantee creation and execution of certain reliable development process. Important attribute of this process is that it can address problems of different involving discipline and is cost effective.

Newly derived evaluation criteria provides certain definite grounds for conducting analysis like simplicity which make it possible to investigate, whether the language semantics are understandable by different stakeholders or not. Similarly preciseness refers to clear-cut translation of language semantics into mathematical expressions. In the same way completeness demands that SysML should support verity of interest and activities. In short, evaluation criteria provide definite dimensions under which analysis of language is conducted. This makes results of study more focused and reliable.

Table10, clearly shows that after applying derived evaluation criteria [35] on SIMILAR [9, 34] it give the impression that SysML can support different system engineering activities across system life cycle i.e. SIMILAR. For instance SysML provide a good support requirements engineering. Now it is possible for system engineers to model requirements and their respective relationship explicitly. Further more requirement diagram also facilitates verification and validation activities. On the other hand it is ultimate aim of system engineering to achieve maximum customer satisfaction by fulfilling customer requirements. SysML also support system engineering in dealing with complexities. Block definition diagram of is good consolation for system engineer to handle complexities. Further more parametric model can support different critical activities like performance assessment, making trade-off and implementing constraints.

After executing SIMILAR [9, 34] in conjunction with SysML under derived criteria[35], it is also found that language not only can provide good support to system engineering activities at different phases of SIMILAR[9, 34] but making process more visible. For example in initial phases requirement diagram is seems to be very useful for requirement engineering. This diagram also offers support for traceability which it possible to track requirements against design elements. Furthermore test case can be easily defined in requirement diagram which shows that V & V activities are very well supported by SysML. Now engineers can address complexity related problem with Block diagram through which they not only break system into different sub-system but also have close look into internal structure of components. Performance analysis is another important concern for system engineers, by introducing parametric model SysML shows good support for performance measurement and implementation of different quality requirements. In short, SysML can make system engineering process more manageable and controllable.

Criterion	**SIMILAR Phases**							**Remarks**
	State the problem	Investigate alternatives	Model the problem	Integrations	Launch the system	Assess performance	Re-evaluation	
Simplicity	H	L	M	M	H	H	H	The language shows good support for requirement engineering, shows some weaknesses for analysing alternative less support for molding business interests in selecting alternatives.
Explicitness	M	L	H	H	H	H	M	SysML generally offer explicit set of notation. As it shows fewer capabilities in analysing market situation. So it become lee explicit in second phase but then due to the presence of block diagram it highly explicit in later phases.
Preciseness	H	H	H	H	H	H	H	Due to the existence of parametric model in SysML, language shows strong Preciseness.
Completeness	H	M	M	M	M	H	M	As SysML is a domain specific language i.e. only supports system engineering related interested and activities. The clearly shows that SysML is not a general purpose language
Scalability	H	H	H	H	H	H	H	It is obvious from result that language can support development of large scale complex system very well which is one of the important aim of system engineering.

Flexibility	M	M	M	M	H	H	H	As SIMILAR progress language shows more support for change management. It means configuration management issue for complex industrial system can be address more effectively with SysML.

Table10. Summarized Evaluation Table

7.2 Validity Threats

Every research study carries some threats to its validity. It is always difficult to judge these threats. Actually validity refers “whether a study measure what it claims to measure”. [46, 47, 48]
Validation threats for this thesis will be observed in light of following four types of validation i.e. internal validity, external validity, construct validity and conclusion validity. [46, 47, 48, 49, 50]

7.2.1 Internal validity

Internal validity is based on cause and effect relationship. Ideal situation for internal validity is that no external variable should affect the result of study. In context of thesis following threats to internal validity will be discussed. [46, 47, 48, 49, 50]

7.2.1.1 Maturation threats

Different internal changes in subject always create maturation threats. SysML is new technology for industrial circles. Due to changing industrial requirements and trends, changes can be expected in internal structure of SysML.

7.2.1.2 Testing threats

Previous execution of same test will generate testing threats. Derived evaluation criteria is playing pivotal role in this study. As this criteria is based on Friedenthal and Roger Burkhart criteria which is already used for assessment of different capabilities of modeling language. This factor is carrying testing threats for this study.

7.2.1.3 Selection threats

Selection threat will come on to surface when no random method is adopted for selection of conditions or criteria. Selection threat to our study is high as there is no random procedure is adopted in the selection of case study, SIMILAR. Further more while driving evaluation criteria different points have been selected with out any random method.

7.2.2 Construct Validity

Construct validity is actually the measurement of how well theory is translated into observation. It is obvious from above mentioned definition that in construct validity there two significant areas of concern i.e. theory and observation. Where theory refers to mental sketch of researcher i.e. what he trying to articulate? Where observation deals with what is actually happening in real world? [46, 47, 48, 49, 50]

For this study threats to constructs validity exit in formulation of support level of SysML for each point of evaluation criteria. As these levels are established in light of experiences, study, it may possible that there exit certain flaws and deficiencies in determining appropriate level i.e. high, medium, low against points of criteria.

7.2.3 External Validity

External validity deals with the process of generalization i.e. moving from sample to population. Ideally process of generalization should be free from any bias. We generalizing result on basis of one case study external validity of this study seem to be low. [46, 47, 48, 49, 50]

7.2.4 Conclusion Validity

Consideration of threats to conclusion validity very important because these threats may lead to wrong conclusion. In general following two types of errors may be encountered in developing relationships in the research study. [46, 47, 48, 49, 50]

1. There is a relationship but researcher concludes that there is no relationship.
2. Researcher concludes that there is a relationship but in fact there is no relationship.

Second point seems to be important for this study. In current study we relate derived evaluation criteria with SysML. Original criteria of Friedenthal and Burkhart [35] were actually general evaluation criteria for modeling languages. As the SysML is new language this criteria may not be suitable for SysML. This fact becomes a threat to conclusion validity for this study.

7.3 Future Work

It is obvious that system engineering application have a very specific mission statement, so it is necessary to lay down strong criterion for verification and validation, figures of merit, risk analysis, trade-off, interface definition and sensitivity analysis.

To get world wide acceptance is one most important future task for SysML. For this purpose SysML must show extra ordinary potential, so different vendor may invest on SysML tool support. In order to make this study time feasible, three major areas of SysML have been considered. It would be better in future to include other areas as well.

Support for verification and validation in SysML is an other important area of future research because need of strong V& V mechanism is very necessary critical systems. Although SysML have good support for reuse but language must enhance its capabilities in this regard should be increased to attract system engineering community and other groups. [45]

8 REFERENCE:

[1] Object Management Group "*OMG SysML Specification*" Dated 2006-05-03

[2] Object Management Group www.omg.org, Retrieved on February 12, 2007

[3] The OMG systems Modeling Language (OMG SysML™) www.omg**sysml**.org/, Retrieved on February 12, 2007

[4] SysML - Open Source Specification Project www.sysml.org, Retrieved on February 12, 2007

[5] IBM Rational Rose Real Time "*A guide for evaluation and review*", Jun 2003 http://www-106.ibm.com/developerworks/rational/library/622.html, Retrieved on February 12, 2007

[6] Bruce Powel Douglass *"Real Time UML: Advances in The UML for Real-Time Systems"*, Third Edition Addison Wesley February 20, 2004

[7] Alan Moore "*The (continuing) Evolution of UML for Real Time Systems Development*" Dedicated Systems Magazine Copyright 2000

[8] Department of Defense *"System Engineering Fundamentals"* System Management College Copyright © 2001 Defense Acquisition UniversityPress Fort Belvoir , Virginia 22060-5565

[9] International Council of System Engineering (INCOSE), http://www.incose.org/practice/fellowsconsensus.aspx, Retrieved on February 12, 2007

[10] R.H. Thayer, "Software System Engineering: A Tutorial," *SoftwareEngineering Volume 1: The Development Process,* 2nd ed., R.H. Thayer and M.Dorfman, eds., IEEE CS Press, Los Alamitos, Calif., 2002, pp. 97-116

[11] International Council of System Engineering (INCOSE), http://www.incose.org Retrieved on February 12, 2007

[12] Frank R. Parth **"***Systems Engineering Drivers in Defense and in CommercialPractice***"** Copyright © 1998 John Wiley & Sons, Inc. CCC 1098-1241/98/010082- 08. Dated February 15, 1998

[13] Stephan J. and Peter A.F *"Software System Engineering: the case for a new discipline"* Software Engineering Journal.

[14] Pressman, R.S. *"Software engineering: a practitioner's approach"* McGraw-Hill Book Co., New York, 1997 4th ed. ISBN 0-07-052182-4.

[15] James Rambaugh, Booch, Ivar Jacobson, "*The Unified Modelling Language Reference Manual*", Addision Wesley Longman Inc.

[16] Sanford Friedenthal, Alan Moore, Rick Steiner *"OMG Systems Modeling Languages Tutorial"*, 11 July 2006

[17] SysML Forum, http://www.sysmlforum.com/FAQ.htm , Retrieved on February 12, 2007

[18] Sanford Friedenthal, Alan Moore, Rick Steiner *"SysML tutorial base line to INCOSE"* copyright © 2006 by Object management Group.

[19] Erik Herzog, Asmus Pandikow "*SysML Assessment*" Syntell AB, SE 100 55 Stockholm, Sweden

[20] Mathew.H *"The UML™ for Systems Engineering Initiative"* International Council of System Engineering, UK Chapter

[21] James Rambaugh, Booch, Ivar Jacobson, "*The Unified Modelling Language Reference Manual*", Addision Wesley Longman Inc

[22] Systems Engineering Domain Special Interest Group http://syseng.omg.org/, Retrieved on May 29, 2007.

[23] Rick Steiner "*System Modelling Language and Mission Assurance*" Raytheon Technology Today issue 1, 2006.

[24] Mathew Hause and Alan Moore *"The System Modelling Language"* Artisan software Tools, Copyright 2006.

[25] Stephanie White, Murray Cantor, Sanford Friedenthal, Kobryn, Byron Purves, "*Extending UML from Software to Systems Engineering"* Proceedings of the 10 the IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03) © 2003 IEEE

[26] Dr Jon Holt and Simon Perry *"SysML: describing the system"* IET Information Professional | August/September 2006

[27] Wei S, Xue-Shan L, Yao-Hong Z *"Research On Simulation Based C4isr System Integrated Design Technology"* Key Laboratory of C4ISR Technology, National University of Defense Technology, Changsha 410073, China. Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006

[28] Yves V, Wim D "*SysML and Systems Engineering Applied to UML-Based SoC Design"*Katholieke Universiteit Leuven,EE Department (ESAT-MICAS), In: Proc. 2nd UML-SoC Workshop at 42nd DAC, Anaheim (CA), USA, 2005

[29] Matthew H,Francis T, Alan M "*Inside SysML*" IEE computing and Control engineering, Augest/sepembter 2005.

[30] Yves V, Wim D *"UML 2 and SysML: an Approach to Deal with Complexity in SoC/NoC Design"* Dept. of EE, ESAT-MICAS, Katholieke Universiteit Leuven, B-3001 Leuven, Belgium. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)

[31] Ipek Ozkaya "*Representing Requirement Relationships*" Software Engineering Institute, Carnegie Mellon University, First International Workshop on Requirements Engineering Visualization (REV'06) © 2006 IEEE

[32] Conrad B **"***SysML and UML 2 Support for Activity Modeling"* U.S. National Institute of Standards and Technology, 100 Bureau Drive, Stop 8263, Gaithersburg, Systems Engineering, Vol. 9, No. 2, © 2006 Wiley Periodicals, Inc.

[33] Laurent B *"An Overview of the Systems Modeling Language for Products and Systems Development"* IBM Research Division,T.J. Watson Center and Tokyo Research Lab. © Copyright IBM Corp 2006

[34] Terry Bahill A, Frank F. Dean "*What Is Systems Engineering? A Consensus of SeniorSystems Engineers*" Systems and Industrial Engineering, University of Arizona. Tucson, AZ 85721-0020. Sandia National Laboratories, Albuquerque, NM 87185. Copyright © 2006 by Terry Bahill and Frank Dean

[35] Friedenthal S. A, Burkhart R *"Extending UML™ from Software to Systems"* OMG SE DSIG Chair /INCOSE Liaison to the OMG,Lockheed Martin Corporation,3201Jermantown Road, Fairfax, VA 22030,Deere & Company, OneJohn Deere Place,Moline, IL 61265.

[36] Douglass B.P *"SysML – The Systems Modeling Language"* Chief Evangelist, I-Logix. © I-Logix. bdouglass@ilogix.com, Retrieved on February 12, 2007

[37] Doyle C, Lloyd R "*Application Lifecycle Managementin Embedded Systems Engineering"* ESE Magazine/Application Software Development www.applicationsoftwaredeveloper.com,Retrieved on February 12, 2007

[38] Aizenbud-Reshef N , Nolan B. T, Rubin J, Shaham-Gafni Y "*Model traceability"* IBM SYSTEMS JOURNAL, VOL 45, NO 3, © Copyright 2006 by International, Business Machines Corporation

[39] McKorkle S, *"Managing embedded systems complexity with SysML"* Telelogic http://www.embedded.com/showArticle.jhtml?articleID=188500902 , Retrieved on February 12, 2007

[40] Conrad Bock "*Systems engineering in the product lifecycle"* US National Institute of Standards and Technology, Manufacturing Engineering Laboratory, 100 Bureau Drive,

Stop 8263,Gaithersburg, MD 20899 8263, USA. *Int. J. Product Development, Vol. 2, Nos. 1/2, 2005*

[41] Hause M, Thom F, and Moore A, *"An overview of Systems Modeling Language"* http://www.embedded.com/showArticle.jhtml?articleID=175002582, Retrieved on May 10, 2007.

[42] International Council of System Engineering (INCOSE), http://www.incose.org/practice/whatissystemseng.aspx , Retrieved on May 10, 2007.

[43] Hause M and Moore A *"The System Modeling Language"* Artisan software Tools, Copyright 2006.

[44] System Modelling Language forum http://www.sysmlforum.com/FAQ.htm Retrieved on May 10, 2007

[45] Willard.B *"UML for system engineering"* Lockheed Martin Aeronautics company GA 30063 USA.

[46] http://www.holah.karoo.net/validity.htm retrieved on September 25, 2007.

[47] http://www.socialresearchmethods.net/kb/considea.php retrieved on September 25, 2007.

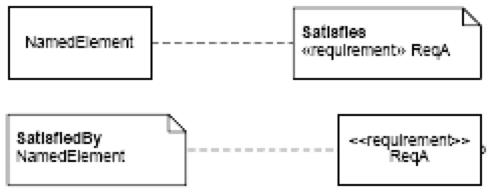
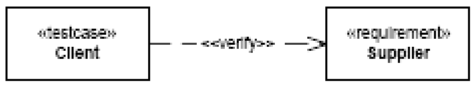
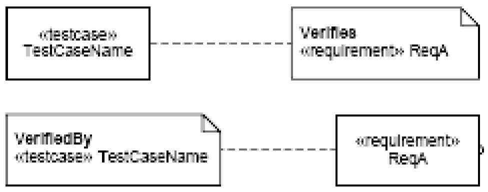
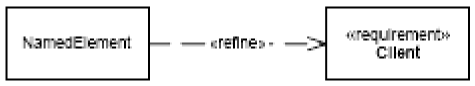
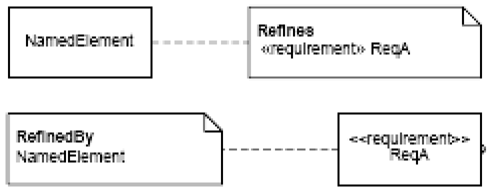
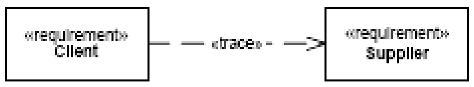
[48] http://www.socialresearchmethods.net/kb/external.php retrieved on September 25, 2007.

[49] http://www.socialresearchmethods.net/kb/convdisc.php retrieved on September 25, 2007.

[50] http://www.une.edu.au/WebStat/unit_materials/c2_research_design/design_experimental.htm retrieved on September 25, 2007.

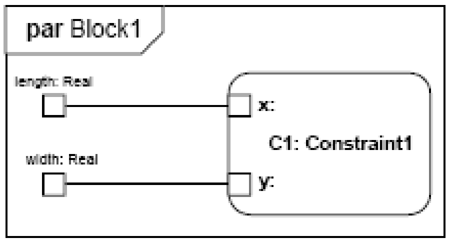
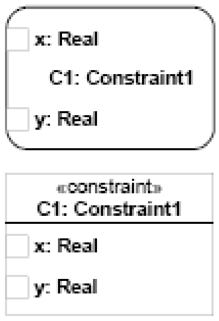
APPENDIX I

SysML notations for Requirement Diagram
Sources: Object management Group" *OMG SysML Specification" Dated* /2006-05-03

Path Type	Concrete Syntax	Abstract Syntax Reference
SatisfyCallout	 NamedElement; Satisfies «requirement» ReqA; SatisfiedBy NamedElement; <<requirement>> ReqA	SysML::Requirements:: Satisfy
Verify Dependency		SysML::Requirements:: Verify
VerifyCallout	 «testcase» TestCaseName; Verifies «requirement» ReqA; VerifiedBy «testcase» TestCaseName; «requirement» ReqA	SysML::Requirements:: Verify
Refine Dependency		UML4SysML::Refine
RefineCallout	 NamedElement; Refines «requirement» ReqA; RefinedBy NamedElement; <<requirement>> ReqA	UML4SysML::Refine
Trace Dependency		UML4SysML::Trace

APPENDIX II

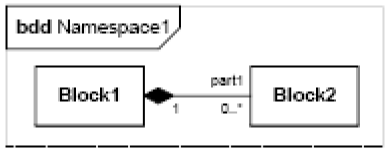
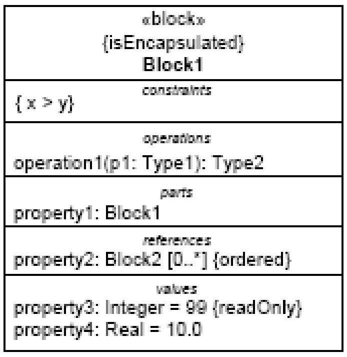
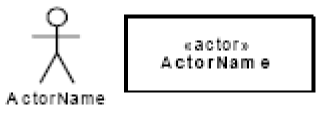
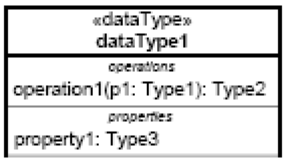
SysML notations for Parametric Diagram
Sources: Object management Group" *OMG SysML Specification" Dated* /2006-05-03

Element Name	Concrete Syntax Example	Metamodel Reference
ParametricDiagram	 par Block1 length: Real width: Real x: C1: Constraint1 y:	SysML::Constraint-Blocks::ConstraintBlock SysML::Blocks::Block
ConstraintProperty	 x: Real C1: Constraint1 y: Real «constraint» C1: Constraint1 x: Real y: Real	SysML::ConstraintBlocks:: ConstraintProperty

APPENDIX III

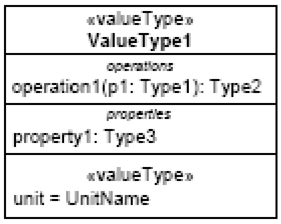
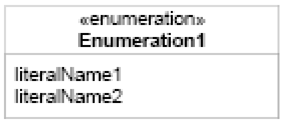
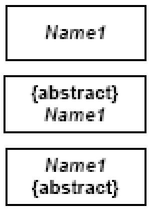
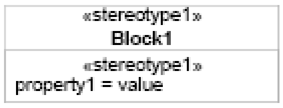
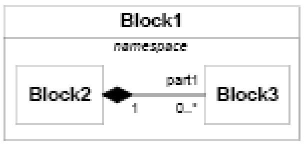
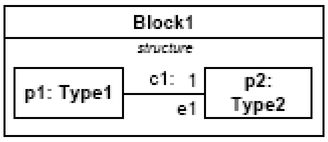
SysML notations for Block Diagram

Sources: Object management Group" *OMG SysML Specification" Dated* /2006-05-03

Element Name	Concrete Syntax Example	Abstract syntax Reference
BlockDefinition Diagram		SysML::Blocks::Block UML4SysML::Package
Block	 «block» {isEncapsulated} Block1 constraints { x > y} operations operation1(p1: Type1): Type2 parts property1: Block1 references property2: Block2 [0..*] {ordered} values property3: Integer = 99 {readOnly} property4: Real = 10.0	SysML::Blocks::Block
Actor		UML4SysML::Actor
DataType	 «dataType» dataType1 operations operation1(p1: Type1): Type2 properties property1: Type3	UML4SysML::DataType

APPENDIX IV

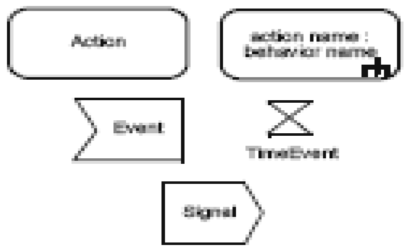
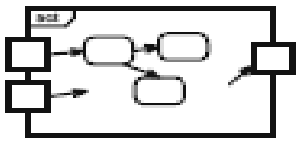
SysML notations for Block Diagram
Sources: Object management Group" *OMG SysML Specification" Dated* /2006-05-03

Element Name	Concrete Syntax Example	Abstract syntax Reference
ValueType	 «valueType» ValueType1 operations operation1(p1: Type1): Type2 properties property1: Type3 «valueType» unit = UnitName	SysML::Blocks::ValueType
Enumeration	 «enumeration» Enumeration1 literalName1 literalName2	UML4SysML::Enumeration
AbstractDefinition		UML4SysML::Classifier with isAbstract equal true
StereotypeProperty Compartment	 «stereotype1» Block1 «stereotype1» property1 = value	UML4SysML::Stereotype
Namespace Compartment	 Block1 namespace Block2 part1 1 0..* Block3	SysML::Blocks::Block
Structure Compartment	 Block1 structure p1: Type1 c1: 1 e1 p2: Type2	SysML::Blocks::Block

APPENDIX V

SysML notations for Activity Diagram

Sources: Object management Group" *OMG SysML Specification" Dated* /2006-05-03

Node Name	Concrete Syntax	Abstract Syntax Reference
Action, CallBehaviorAction, AcceptEventAction, Send-SignalAction	 Action action name : behavior name Event TimeEvent Signal	UML4SysML::Action, UML4SysML::CallBehaviorAction UML4SysML::AcceptEventAction UML4SysML::SendSignalAction
Activity		UML4SysML::Activity