A STUDY ON CONCEPTUAL MODELING IN SIMULATION SYSTEMS: AN EXTENDED METHODOLOGY FOR KAMA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BANU E. AYSOLMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

DECEMBER 2007

Approval of the Graduate School of Informatics

_____

Prof. Dr. Nazife Baykal

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Yasemin Yardımcı

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Onur Demirörs

Supervisor

Examining Committee Members

Prof. Dr. Semih BİLGEN (METU, EEE) _____

Assoc. Prof. Dr. Onur DEMİRÖRS (METU, IS) _____

Assist. Prof. Dr. Aysu Betin CAN (METU, IS) _____

Assoc. Prof. Dr. Ali DOĞRU (METU, CENG) _____

Assoc. Prof. Dr. Veysi İŞLER (METU, CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this wok.

Name, Last name : **Banu E. Aysolmaz**

Signature : _____

ABSTRACT

A STUDY ON CONCEPTUAL MODELING IN SIMULATION SYSTEMS:
AN EXTENDED METHODOLOGY FOR KAMA

Aysolmaz, Banu E.
M.S., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Onur Demirörs

December 2007, 190 pages

Conceptual modeling is considered to be essential in simulation development activities. However, there are only a few research studies on how to develop conceptual models. One of the important and comprehensive approaches is the methodology developed under the leadership of METU Modeling and Simulation Center (MODSIMMER) for Turkish Armed Forces. The project suggests a methodology to develop mission space conceptual models (GUKAM), and provides a Conceptual Model Development Tool for C4ISR M&S activities, which is named as KAMA-C4ISRMOS. KAMA methodology is developed to utilize conceptual models in requirements collection and analysis activities.

Two improvement opportunities observed in KAMA approach are that, there are no methodologies defined to develop simulation space conceptual models; and although most approaches emphasize the importance of conceptual model to be used in design activities, no explanations are provided on how to do it. This thesis aims to suggest an extended KAMA methodology that, besides original KAMA properties, provides a method to develop simulation space conceptual model, and provides a guide to use

conceptual model to develop design. To evaluate the suggested methodology, a case study is conducted on a synthetic environment project. In this way, implementation of the methodology on another simulation domain is depicted. Developed mission space and simulation space conceptual models and design artifacts are evaluated, and the effects of conceptual models on simulation development life cycle are discussed.

Keywords: Conceptual Model, Simulation, Design, KAMA

ÖZ

SİMULASYON SİSTEMLERİNDE KAVRAMSAL MODELLEME ÜZERİNE BİR ÇALIŞMA:
KAMA İÇİN GENİŞLETİLMİŞ BİR METODOLOJİ

Aysolmaz, Banu E.
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yoneticisi: Doç. Dr. Onur Demirörs

Aralık 2007, 190 sayfa

Kavramsal modelleme simülasyon geliştirme faaliyetleri için zorunlu görülmektedir. Ancak, kavramsal modellerin nasıl geliştirileceği üzerine sadece birkaç çalışma vardır. Önemli ve kapsamlı yaklaşımlardan biri ODTÜ Modelleme ve Simülasyon Merkezi (MODSİMMER) liderliğinde, Türk Silahlı Kuvvetleri için geliştirilen metodolojidir. Proje, görev uzayı kavramsal modelleri (GUKAM) geliştirimi için bir metodoloji önermekte; ve C4ISR modelleme ve simülasyon aktiviteleri için KAMA-C4ISRMOS isimli "Kavramsal Model Geliştirme Aracı" oluşturmaktadır. KAMA metodolojisi, kavramsal modelleri gereksinim toplama ve analiz faaliyetlerinde kullanmak için geliştirilmiştir.

KAMA yaklaşımında görülen iki gelişme imkanı, simülasyon uzayı kavramsal modellerinin geliştirilmesi için tanımlı bir metodoloji olmaması; ve bir çok yaklaşımın kavramsal modelin tasarım faaliyetlerinde kullanılmasının önemini vurgulamasına rağmen, nasıl kullanılacağına ilişkin herhangi bir açıklama olmamasıdır. Bu tez çalışması, KAMA'nın özelliklerinin yanında, simulasyon uzayı

kavramsal modelleri geliştirmek için bir metod ve kavramsal modeli tasarımda kullanmak için bir rehber sağlar. Önerilen metodolojiyi değerlendirmek için, bir sentetik çevre projesi üzerinde vaka analizi gerçekleştirilmiştir. Bu şekilde metodolojinin başka bir simülasyon alanında uygulaması gösterilmiştir. Bu çalışma sonucu olarak geliştirilmiş görev uzayı ve simülasyon uzayı kavramsal modelleri ve tasarım ürünleri değerlendirilmiş, ve kavramsal modelin simülasyon geliştirme yaşam döngüsündeki etkileri karşılaştırılmıştır.

Anahtar Kelimeler: Kavramsal Modelleme, Simulasyon, Tasarım, KAMA

ACKNOWLEDGMENTS

Firstly, I would like to express my thanks to my supervisor, Onur Demirörs, for his patient support and guidance through my studies. He has been motivating even in hard times; and always came with a solution to the problems. I have been comfortable with and relying on him about all technical and personal issues.

I want to express my appreciation to KAMA development team for their support. I especially thank to Utkan Eryılmaz, whom I called every now and then to understand KAMA and discuss my studies. I also thank the development team of my case study project, who sincerely evaluated the proposed methodology. I appreciate their belief in new methodologies to enhance simulation development.

Special thanks to my husband tolerating my feigned reluctance all along, and relaxing me at my nervous times. I would more like to feel that this is “ours”, still, it would be too hard without you.

Thanks to my brother for listening my never ending concerns about my thesis. Lastly, I thank my mother and father, for supporting me through all my long education life and never doubting on what I want to do. I know I stressed my dad so much on this study. Trust me, I never thought of quitting. But your gentle support and worry was always the motivation for me. Yet, probably, this study can be spelled as the end of my education life; but I always need the two of your’s support*.

* Son olarak anneme ve babama, beni uzun eğitim hayatım boyunca destekledikleri ve ne yapmak istediğim hakkında hiç şüphe etmedikleri için, teşekkür ediyorum. Biliyorum ki bu çalışmada babamı çok strese soktum. İnan bana, bırakmayı hiç düşünmedim. Ama senin nazik desteğin ve endişen benim için her zaman motivasyondu. Artık, herhalde, bu çalışma eğitim hayatımın sonu denebilir, fakat ikinizin desteğine her zaman ihtiyacım var.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS / ACRONYMS

C4ISR	: Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CASE	: Computer Aided Software Engineering
CGF	: Computer Generated Forces
CH	: Command Hierarchy (diagram type in CM)
CM	: Conceptual Model
CMM	: Capability Maturity Model of SEI
CMMS	: Conceptual Models of the Mission Space
CSS	: Common Semantics and Syntax
DB	: Database
DCMF	: Defence Conceptual Modeling Framework
DIF	: Data Interchange Formats
DMSO	: Defense M&S Office
EATI	: Entities, Actions, Tasks, Interactions
EO	: Entity Ontology (diagram type in CM)
ER	: Entity Relationships (diagram type in CM)
ES	: Entity State (diagram type in CM)
Euclid	: European Co-operation for the Long-term in Defence
FCM	: Federation Conceptual Model of FEDEP and SEDEP
FDMS	: Functional Descriptions of the Mission Space
FEDEP	: HLA Federation Development and Execution Process
FOM	: Federate Object Model of HLA
FMS	: Flight Mission Simulator
GIS	: Geographic Information System
GUKAM	: Short for “Mission Space Conceptual Models” in Turkish
HLA	: High Level Architecture
HLD	: High Level Design
IS	: Information Systems
ISO/IEC 12207	: Standard for Information Technology Software Life Cycle Processes
ISR	: Intelligence, Surveillance and Reconnaissance
JCMMS	: Joint CMMS
JSIMS	: Joint Simulation System
JWARS	: Joint Warfare System
KAMA	: Short for conceptual modeling project developed for Turkish Armed Forces
LOS	: Line Of Sight
M&S	: Modeling and Simulation

MDA	: Model Driven Architecture
MOF	: Meta Object Facility
MS	: Mission Space
MisSp	: Mission Space (diagram type in CM)
OOD	: Object Oriented Development
OS	: Organization Structure (diagram type in CM)
PIM	: Platform Independent Models
PO	: Prime Opponent
RPG	: Recommended Practices Guide
RTP	: Research and Technology Programme
SAF	: Semi Automated Force
SE	: Synthetic Environment
SDLC	: Software Development Life Cycle
SEDE	: SE Development Environment
SEDEP	: Synthetic Environment Development and Exploitation Process
SS	: Simulation Space
SME	: Subject Matter Expert
SOM	: Simulation Object Model of HLA
US DoD	: U.S. Department of Defense
V&V	: Verification and Validation
VV&A	: Verification, Validation and Accreditation
WF	: Work Flow (diagram type in CM)
YEROOS	: Yet another project on Evaluation and Research on Object-Oriented Strategies

CHAPTER 1

INTRODUCTION

Implementation of software in defense industry mostly includes modeling and simulation fields, which are usually complex in nature. This complexity requires well-defined processes to minimize the risk and to increase the chance of developing systems that meets user requirements within defined budget and schedule.

Software development processes include domain analysis, user and system requirements analysis, system design, implementation and testing activities. Depending on the type of process model used; like waterfall, spiral, evolutionary or agile development models, the order and implementation of these activities may change; still all models contain at least the activities stated above.

Modeling and Simulation field realizes the importance of conceptual modeling in simulation development. Conceptual modeling activity is usually placed between requirements analysis and design activities in the life cycle model. Its aim is to organize user needs as a formal model for understanding the required system in detail and to serve as an input for consequent activities. Other than providing good understanding of the system, being a communication device between developer and user, and in this way, helping in flawless requirements development; conceptual model is also utilized in design activities as a direct input, and even used in development and V&V activities, to minimize errors in latter stages of development.

Although there are a lot of studies in the literature emphasizing the importance of conceptual modeling in simulation development, there are not many well-defined methodologies on how to develop conceptual models. There are a few approaches that define conceptual modeling and suggest guidelines on how to develop and use conceptual models. One of the important approaches is the methodology developed by the consortium of METU Modeling and Simulation Center (MODSIMMER), Meteksan Sistem A.Ş. and Bilgi Grubu Ltd.Şti. for Turkish Armed Forces. The project has proposed a methodology to develop mission space conceptual models (GUKAM); and provides a Conceptual Model Development Tool for C4ISR M&S activities, which is named as KAMA-C4ISRMOS. The project is shortly named as "KAMA". KAMA methodology is developed to utilize conceptual models in requirements collection and analysis activities. It is mainly targeted for C4ISR domain of military simulations. This thesis study aims to suggest an extended KAMA methodology that, besides original KAMA properties, provides a method to develop simulation space conceptual model, and provides a guide to use conceptual model to develop design. Also, the methodology is targeted to simulation domains other than C4ISR, by applying it in a different domain, namely synthetic environments for simulation systems.

In the following parts; modeling, simulation, conceptual modeling concepts will be defined, and problem statement, scope and outline of this study will be provided.

1.1 Simulation and Conceptual Modeling

Modeling can be defined as abstraction of reality for the concerned domain. Simulation is defined as "the imitative representation of the functioning of one system or process by means of the functioning of another [1]".

Modeling and simulation are usually confused with each other. Model is a static abstract representation of a system with its own assumptions and limitations. Simulation is "dynamic, digital implementation of a model over time that generates an artificial history of modeled systems [2]". With the same point of view, one may

observe that, to be able to develop a simulation, one must first distill the entities and relationships of the related domain into a model. That will be the most intuitive reason for discussions in the following chapters stating that developing a conceptual model is a must for simulation systems.

Conceptual model (CM) can be defined as "an abstract representation of something generalized from particular instances [3]". There are different perspectives for CM from knowledge engineering and cognitive science. These perspectives imply that "CM involves constructing representations of human knowledge [4]". Robinson defines conceptual modeling as "the abstraction of a model from a real or proposed system, which includes simplification of reality [5]".

Simulation conceptual modeling aims to provide a good understanding of problem domain. The term "conceptual modeling" is used in many fields. This study deals only with simulation conceptual modeling. For that reason, from now on, the name "Conceptual Model (CM)" will be used for "Simulation Conceptual Model" in this study.

Various definitions are suggested for CM. Pace defines CM as "translating modeling requirements into detailed simulation specifications (and associated simulation design) which fully satisfy requirements [6]".

Glossary of M&S terms created by US DoD, which also follows approach of DIS community of 1990s, defines CM as "the agreement between the simulation developer and the user about what the simulation will do [7]".

Robinson defines CM as "a non-software specific description of the simulation model that is to be developed, describing objectives, inputs, outputs, content, assumptions and simplifications of the model [5]".

FEDEP defines CM as "an abstraction of real world that serves as a frame of reference for federation development by documenting simulation-neutral views of important entities and their key actions and interactions [8]".

To sum up, it is mostly agreed that CM has the following properties;

- Conceptual modeling activity is iterative and repetitive through all development cycle
- CM is a simplified representation of real system
- CM is independent of model code or software
- "Perspectives of both user and developer are taken into consideration [5]"

It is important not to confuse conceptual model and simulation model. A CM is an abstract model of reality which is platform independent. However, simulation model is "computerized version of a CM which is platform dependent; and a CM can be implemented by multiple simulation models [9]".

It is accepted by researchers that CM development is a critical phase within simulation development process, which is connected with other phases with strong input and output relations. Conceptual modeling is placed between requirements analysis and design activities in development process. Domain knowledge and requirements are used as inputs for CM, and requirements and design are developed as outputs, by using CM. The placement of CM in SDLC enhances software development process by means of strong input/output relations between steps. Pace's statement that "requirements and CM development go iteratively, they affect and derive from each other; and they are also finalized iteratively [4]" emphasizes this relation. Zeigler states that "modeling is concerned with relations between real systems and models, while simulation is referred to relations between computers and models [3]". This point of view also supports that CM stands as an independent artifact between real world and developed system.

CM is divided into two major parts, mission space and simulation space. Mission space (MS) conceptual model includes the models of the concepts in military domain of the related simulation. It includes real life operational considerations and domain knowledge of entities, relations and actions. Because of its properties, MS models are rather in relation with and used in requirements analysis phase. Simulation space (SM) conceptual model includes simulation concepts, functional and operational

capabilities of simulation and considers objectives, assumptions and constraints of the simulation system. SS model is developed using requirements and usually after development of MS model. Because of its structure, SS model is mostly utilized in design activities. Lacy, in a DMSO meeting, tried to clarify the perspectives of the group on CM and summarized the placement of MS and SS conceptual models in simulation development life cycle as in ***Figure 1***, which summarizes the discussions above [24].

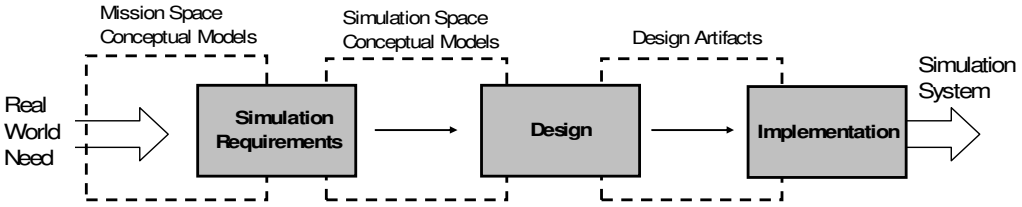


Figure 1: Place of MS and SS Conceptual Model in SDLC

1.2 Problem Statement

There are different discussions on and approaches to conceptual modeling in literature; which will be discussed in latter sections. Considering the existing approaches, it is observed that there is no widely accepted CM development methodology. Some approaches; like CMMS and FEDEP; emphasize the importance of CM, provide some explanations on the area and give some guidelines on how to define CM. However, they do not provide detailed descriptions on how to compose and use CM. There are a few CM implementations developed for specific projects, but these implementations do not usually include methods applicable for different simulation systems; in contrast, they are just specific to projects they are applied.

As mentioned, conceptual modeling applications in literature focus on the specific area they are intended to. Even KAMA, which this study is based on, is mainly

targeted for simulation systems that belong to C4ISR domain. There is a lack of conceptual modeling methodology that covers different areas of military domain.

A deficiency observed in conceptual modeling field is that; almost all of the current approaches focus on mission space CM development; but there is no method for developing simulation space CM. Although many approaches state that SS model is an essential part of CM; such a method does not exist in current approaches. Another deficiency is lack of any guidelines on how to utilize CM in software design activities. Although most approaches emphasize that CM is an input for design activities, there are no studies providing explanation on how CM can be benefited from during design.

In summary, main problems of conceptual modeling of military simulations domain can be grouped in three headings. Firstly, there is not a widely accepted CM development methodology that explains CM development process thoroughly and that can be used for different types of military simulations. Secondly, there is no methodology explaining how to develop simulation space CM. And lastly, there are no guidelines explaining how to utilize CM as an input in design activities.

1.3 Scope of the Thesis

It is mentioned in previous section that existing conceptual modeling approaches do not provide comprehensive guidelines on how to develop CM. Unlike other approaches, KAMA is a project that has suggested a solid CM development methodology. It clearly defines the elements of CM, the form of CM and how to develop and document CM. It explains CM development process in detail and provides example applications. The suggested method is not just for a specific project, but can be used by different simulation systems. Considering these properties, KAMA covers most deficiencies of existing studies in literature, and offers users a ready to use method. For that reason, this study is based on KAMA.

To solve the three problems determined for conceptual modeling, this study suggests extending KAMA methodology. As specified, this study is based on KAMA CM development methodology, so it already provides benefits and properties KAMA has. KAMA provides a methodology to develop mission space CM, and to utilize CM in requirements collection and analysis activities. Basing the study on KAMA solves the problem of lack of comprehensive mission space CM development methodology.

Considering the fact that there is no methodology to develop simulation space CM, this study aims to cover this issue by suggesting a method for it. By using mission space CM development method as a baseline, a method for developing simulation space CM is proposed. This method is an addition to KAMA as a new aspect in conceptual modeling, but is still in harmony with KAMA.

This study also aims to provide discussions and guide on how to use CM as an input for simulation design activities. For these discussions, suggested MS and SS conceptual model development method is evaluated and if necessary, modifications are made. A guide on how to develop UML design diagrams by using CM as input is explained, which provides developers a high level design before starting detailed design activities.

Despite strong infrastructure of KAMA that can cover many types of systems, it is mainly targeted for simulation systems that belong to C4ISR domain. This study aims to show how to apply extended KAMA method for simulations other than C4ISR domain. Considering this fact, this study depicts the implementation of extended KAMA methodology on another simulations domain, synthetic environment, which cover many aspects of simulation systems. The proposed extended KAMA methodology is evaluated with the synthetic environment implementation.

Case study research is found to be appropriate to evaluate the suggested methodology, as case studies are effective to confirm and test well formulated theories as a qualitative research strategy. A single case study is applied to evaluate

extended KAMA methodology thoroughly. The synthetic environment simulation system case is selected because of its wide coverage to test all aspects of the methodology and present usage of it for different simulation types. In the case study, mission space CM and simulation space CM are developed for the synthetic environment simulation system; and a high level design of the system is developed using CM as an input.

To sum up, this study mainly includes three working areas. First is the extension of KAMA to develop simulation space CM. Second is the extension of KAMA to use CM in design and to provide guidelines on how to develop high level design from CM. Third is a case study research to evaluate suggested extended KAMA methodology and depict usage of the methodology in a different domain.

1.4 Outline of the Thesis

This thesis study is composed of five chapters. Chapter 1 is the introduction chapter, which provides brief information on simulation conceptual modeling field, and explains the scope of the study.

In Chapter 2, literature information on conceptual modeling is provided. First, different usages of conceptual modeling are discussed. Then, current approaches in literature are explained. A few existing CM applications are also provided as examples. A brief introduction to KAMA methodology which this study is based on is also given at this chapter. At the end of the chapter, a discussion of CM approaches and examples is given that evaluates information provided in the chapter.

Chapter 3 is the section that explains proposed CM development methodology. A complete description of the extended methodology is provided, including the basic properties of KAMA. The methodology is explained including information collection, identification of elements, mission space CM development, simulation space CM development, and usage of CM in design. At the end of the chapter, a

discussion on the suggested methodology and inserted extensions with their objectives are provided.

Chapter 4 includes the CM case study research conducted within this study. Design of the case study, research questions, data collection methods and validation criteria are presented. Example diagrams from developed conceptual model and design is presented, in harmony with the methodology explained in former chapter. The findings of the case study are discussed and evaluated; and the answers of research questions are discussed in detail.

The study is finalized in Chapter 5, with a discussion of overall findings. The possible future work in the area is also discussed.

CHAPTER 2

THESIS BACKGROUND

This chapter contains results of literature survey on conceptual modeling field. First, various usages and benefits of CM are discussed and supported with ideas of researchers in the area. Then, the historical and current approaches in the literature are provided. These include both approaches of single researchers and institutions. Also, some CM applications available are explained. In this chapter, information on KAMA project, on which this study is based, is also provided. At the end of the chapter, a discussion that compares and evaluates the approaches is given. The knowledge obtained from this chapter will assist the rest of the studies.

2.1 CM and Usages of CM

Although the benefits of CM found acceptance by many researchers, there is not a widely accepted approach for developing and documenting CM. As Wang Xue-hui states, "the uniform standards do not exist, not only for simulation decomposition into entities and processes, for representation abstraction of the subject simulated, but also for how to describe and document simulation CM [10]". Due to this fact, CM development stays as an art of modeling, and as Borah states, "CM is a living document that grows from an informal description to a formal description to communicate between diverse stakeholders [3]".

CM is a must for every system developed, like a floor plan for building a house. DMSO's Key Concepts of VV&A document declares this situation in the following way. "It is important to recognize that a simulation developer always works from a CM. Even if it is not formally written down, it will exist in the developer's mind [11]". Considering this, obviously, a well defined and written CM will be more beneficial in SDLC, compared to an implicit, erroneous CM that exist only in developer's mind. Including usage in requirements analysis and design phases, CM has many other advantages in development process, as described below.

One of the most important usages of CM is its support in V&V activities. It will be possible to make a solid V&V of the end product, if there is a well-defined CM of intended system. Many researchers has listed the benefits of CM, including contribution of CM in V&V of entire development process; like Pace [12], Chapman [13], Weiner [14].

The importance of CM in V&V is observed in a proposed process maturity model for simulation validation [15]. In this study, S.Y. Harmon has developed a maturity model for validation, resembling CMM. Development and verification of CM is introduced starting from level 2. V&V activities are strongly improved starting from level 3, in which development products are verified against CM.

Pace states that there are two aspects in CM validation. First is the validation of "capability of CM to satisfy simulation requirements"; second is about "capability of CM to support a particular application of the simulation [12]". Similarly, Eryılmaz, Bilgen and Molyer states that CM is used for two kinds of V&V activities in simulation development process; "to validate simulation requirements before finalizing them", and "to verify and validate rest of the system and artifacts that are developed based on CM [16]".

CM is an anchor for rest of development activities in SDLC. Hunt states that, "to support the validation of the overall system, the final outputs of the system should be traceable back to CM [17]". CM is helpful in revealing situations in system that are not observed even in testing phase. Pace states that "CM is the only rational basis for

judging a simulation's capabilities under circumstances other than those specifically tested [6]".

An existing CM for a developed system can be used to evaluate the appropriateness of that system to be reused. Especially, developing standard CM repositories increases the reuse of both CM's, and the end products. Weiner informs that "CM serves as a fitness-for-use evaluation tool for prospective users of the simulation [14]". Personnel with broader responsibility, like master plan developers of U.S. DoD and programme developers, especially emphasize the importance of the reuse benefit of CM for end products. For example, developers of Army Intelligence Master Plan directly relate the success of a simulation to its degree of successful modeling in military operations [18].

As the technology evolves, many simulation systems can work interactively, even with live forces. This results in increased operability and reliability demands among military systems. Pace declares that "a key to achieving such compatibility and reliability in simulation data is the simulation CM because CM is the basis for judgment about the appropriateness of data in distributed simulation [4]".

The study of Andreas Tolk can be given as an example of how CM affects interoperability. He has developed "Levels of Conceptual Interoperability Model" (LCIM) in which five levels of interoperability are defined, starting from no interoperability between two systems (level 0) up to harmonized data (level 4) [19]. The writer strongly emphasizes that, the only way to attain the highest level is usage of CM. The other methodologies used in lower levels like using a common protocol (like object model template), using a common reference model, using standard software engineering methods are not enough on their own to reach the highest level. During modeling, some parts of the real world and its relations are left out and some are included. If the unmodeled relations are needed to provide interoperability, the problem evolves. This situation can be prevented by only developing CM; as it is CM that describes which parts of the real world are modeled under which constraints, and which parts are not.

YEROOS project defines the use of CM in reverse engineering. As the study group states, most of the systems are updated after the systems are started to be used. But documentation is not usually updated; so modifications of the model do not have traceability or reversibility; and most systems are error prone after development phase is completed, because of this fact. A good solution to this problem is to keep up a good CM to increase traceability and reversibility of the system [20].

CM is a tool to provide clear communication between simulation stakeholders. This benefit helps in smooth development activities that have least problems between developers and users. Pace lists the stakeholders as "simulation developer design and implementation personnel (systems analysts, system engineers, software designers, code developers, testers, etc.), simulation users, subject matter experts (SMEs) involved in simulation reviews, and evaluation personnel, such as those involved in VV&A [12]". Robinson also states the importance of CM for providing communication between all parties; "if not, the credibility of the model would be significantly compromised [5]".

As there is a strict relation between simulation requirements and CM, CM development may reveal problems with simulation requirements, like inconsistencies and errors among requirements. Also, "serious holes in the requirements may also be revealed" and "conflicts among different perspectives of the requirements may be determined [12], [21]". In this way, development of CM increases the quality of requirements, which will decrease possible future errors and the costs.

Not only revealing holes in requirements, but also CM is directly used to define requirements. Customer's examining a CM and clarifying requirements accordingly will help determining requirements easily. This development of CM is somehow like prototype development of system. As Gustavson states, "concept of early prototyping, giving the customer something to look at, touch and feel often helps them clarify in their own minds what it is they are really after [22]".

Finally, an important benefit of CM is its usage as an input for design. The usage of CM in design will increase design's correctness, reflect requirements more correctly

in design, and ease the process of design. In DMSO's CM development and validation documentation, it is stated that one of the applications of CM is "as a foundation for design of software and other components [23]". DMSO VV&A RPG also defines CM as the mechanism that is used to transform specifications to design [11]. Although there are many approaches emphasizing the usage of CM in design, there is no clear description of how to do it. This is one of the issues that is discussed and handled in this study.

2.2 History, Current Approaches and Applications

There are various perspectives and approaches on what CM is, that have evolved through time. As stated in Lacy's study with M&S community experts, researchers agree that CM is essential [24]. Most of the approaches try to determine the components of CM, its purpose and its place within the development life cycle.

One of the main problems of CM development is about collection and representation of knowledge. Researchers use knowledge engineering and system engineering tools to model simulation requirements. As Pace states, they focus on "developing formalisms to represent knowledge [4]".

Pace informs that, "in the past, most simulation developments failed to produce explicit CM's; and even contemporary simulation developments do not include a distinct documentation for CM [25]". Gustavson suggest three reasons for the situation and why there is not a well-defined methodology for SCM development [22]. First, developers lack understanding of the benefit they will get by developing a CM. Second, developers do not know what a CM looks like, so they do not know how to capture it. Lastly, there are technology limitations, the standards built up to now are not mature enough to support more advances in CM.

Keeping these issues about CM development in mind, every approach has an importance for the advance in the field. At the following sections, outstanding approaches in literature are provided with brief descriptions.

2.2.1 Early Approaches - Sargent and Davis

Both the researchers Sargent and Davis are early researches to talk about conceptual models and use of them in software development. They have mentioned CM in their studies at about late 80's and early 90's.

Robert Sargent has recommended the use of CM in VV&A activities in simulation development. In his studies in mid 80s; he describes a CM as "the mathematical/logical/verbal representation of the problem entity developed for a particular study"; and the computerized model as "the CM implemented on a computer [26]". He states that CM is developed through an analysis and modeling phase. He suggests the use of CM to verify that computerized model is the correct implementation of the CM.

Paul Davis has also suggested the use of CM for VV&A activities of military simulations. In his report from RAND, he bases his idea on Sargent's studies, and he thinks that it is a good idea to develop a CM preceding programming. He defines CM as "machine and language independent specifications [27]". Although he thinks that separating model design from detailed design is an advantage, he is doubtful that organizations can prepare those models because of lack of discipline.

2.2.2 Pace Approach

Dr Dale Pace has a lot of work on conceptual modeling; and his studies are the basis of DMSO documents. The study he has made in conceptual modeling field has become the mostly accepted issues in the field, and conceptual modeling experts mostly use his definitions as the language to communicate between each other.

He describes conceptual modeling as "simulation developer's way of translating modeling requirements (i.e. what is to be presented by the simulation) into a detailed design framework (i.e. how it is to be done), from which the system (hardware, software, etc.) can be built [4]". Pace states that the information that CM includes is "assumptions, algorithms, characteristics, relationships and data [6]". He describes

CM as "what is to be represented by the simulation (entities, actions, processes, interactions, etc.) [12]".

Pace developed a CM definition and structure that is accepted by DMSO and many other approaches. According to that structure, CM has two components, "Simulation Context" and "Simulation Concept". The structure defined by Pace is provided in ***Figure 2: Conceptual model definition of Pace*** below [4].

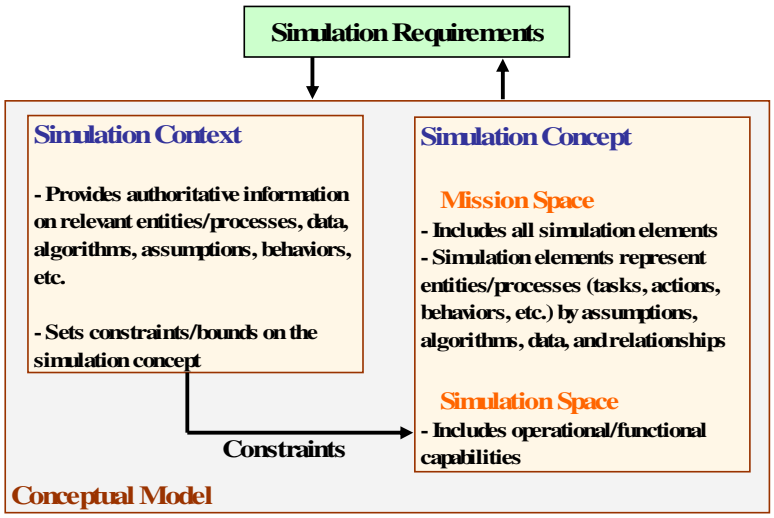


Figure 2: Conceptual model definition of Pace

Simulation context is explained by Pace "to provide authoritative information about the domain which the simulation is to address [12]". Examples of simulation context are laws of physics and general doctrine and tactics for military simulations. Simulation context may be thought as references to domain-related sources from which to obtain related data for the simulation.

Simulation concept is defined by Pace as "the developer's perspective for all application of how the simulation will be built to satisfy user requirements [12]". Constraints and boundaries for simulation concept are defined by simulation context. As depicted in ***Figure 2: Conceptual model definition of Pace*** above, simulation concept includes "mission space" information and "simulation space" information.

Mission space includes things represented in the simulation, which are simulation elements like entities and processes. Simulation space is the place that describes how simulation elements interact with each other.

For the simulation elements, Pace states that they are "information describing concepts for an entity, including assumptions, algorithms, characteristics, relationships, data, etc., and they identify item's states, tasks, events, behaviors, performance, parameters, attributes etc. [4]". An entity may be defined as a small part of another entity, a collection of a few entities, or a composite entity; depending on the detail level of the system.

Simulation space includes the information on how the simulation works. Pace lists such additional information as "control capabilities (pause and restart), data collection and display capabilities, ways of input to simulation (keyboard, touch, internal input), hardware and software limitations, or other implementation issues like implementation of a parallel computing architecture [6]".

Pace clarifies the difference between the "requirements" and the "specifications" in software development. He defines "high level needs statement or objectives for a system" as requirements, and "more detailed requirements" as specifications, which are close to difference between user and system requirements analysis phases [24].

Pace defines four evaluation criteria for a CM to fulfill its objectives as below [4].

- **Completeness:** CM includes "all entities and process" of mission space. CM also includes "all control and operating characteristics" of simulation space.
- **Consistency:** Entities and processes in CM are determined in compatible perspectives, like coordinate systems and units, level of aggregation.
- **Coherence:** All elements of CM have function and potential.
- **Correctness:** CM is appropriate for intended application.

Pace defines a process to provide a guideline on how to develop CM. Steps starts from collecting information, includes decomposition and abstraction of elements, and

finalizes development with determining relationships of elements. These four steps can be briefly explained as follows [12].

- **Collection of authoritative information about simulation context:** As the first step, authoritative information to define simulation context is collected. During CM development process, as the simulation concept is defined, information collection for simulation context will continue. Although it is very important to collect right and complete information, there is no systematic way defined for this activity. Pace suggests usage of knowledge engineering methodologies. Pace also states the importance of usage of a common semantics and syntax, a topic on which more detail will be provided under the approach named "CMMS".
- **Simulation decomposition:** In this step, entities and processes to be represented in the simulation are identified. This is a basic step, as critical decisions like level of fidelity and aggregation are identified. For example, it is decided if a system like a tank will be represented as a single entity; or the barrel of the tank will be represented as a single entity and tank will be represented as a composite entity; or even smaller parts will be represented as the entity. Also, level of representations of human decisions and actions are determined. All these decisions determine the level of detail of CM.

  Pace specifies a six item checklist to decompose simulation into elements. These state that there shall be a specific simulation element for every item (like parameter, attribute, entity, process, etc.) in simulation requirements. This is important for a full traceability of requirements to CM. There shall also be a specific element of potential assessment interest. This is important to be able to use CM in V&V of the simulation. If possible, simulation elements shall correspond to standard and widely accepted decomposition paradigms in problem domain; like algorithms, methodologies or other simulation components. It is also important that elements required for computational considerations (like an element to be used for approximation) shall only be used when it is absolutely necessary. Lastly, there shall be no extraneous elements as every extraneous item is a potential cause of errors.

- **Representational abstraction of elements:** This is the step that simulation elements are developed and classified as one of the element types that Pace defines. Again, as there is no systematic way, knowledge engineering abstraction principles can be used to determine the abstraction. A simulation element is needed for each entity or process identified in the previous step. Here, decisions on level of accuracy, precision, resolution for the representation of entity or processes are made.
- **Determination of relationships among elements:** In this step, relationships among simulation elements are determined so that constraints and boundary conditions are stated in CM. Also, simulation context properties are determined.

Pace thinks that an important problem in conceptual modeling field is that, many simulation developments in the past lack to produce documentation for CM. He proposes scientific paper approach for CM documentation, though he also accepts employing an "implementation-oriented descriptive format like UML [4]". For the content of the scientific paper, he utilizes generic content guidelines provided in the standard ISO/IEC 12207. He determines the following sections for CM: "portion identification, point of contacts, requirements and purpose (for CM), overview (of simulation), general assumptions, identification of elements (states, tasks, actions, behaviors, interactions etc.), identification of algorithms, simulations development plans and summary [4]". He supports developing of more than one view to identify elements; like "functional view" depicting flow of data, "data view" explaining hierarchy and static relations, and "behavioral view" depicting dynamic aspects like states, transitions, interactions [12].

For Pace, CM validation is an important activity to finalize CM efforts. CM validation is assessment or evaluation of CM. Pace explains that a conceptual validation review of a simulation element "determines appropriateness of the representation of that item in the simulation" while validation of a simulation concept "assesses the overall capability of the simulation [12]".

2.2.3 VV&A RPG

US DoD DMSO has a special working group on VV&A activities of M&S systems. They name their study as "VV&A RPG". While working on how to conduct VV&A effectively, they also recognize conceptual modeling as a tool in VV&A activities, and they work on conceptual modeling as a special topic within their studies.

Dr.Pace's work has been passed on to VV&A RPG, so there are similarities between Pace's studies and VV&A RPG. RPG uses Pace's definitions of simulation context, simulation concept, MS, SS and simulation elements as they are. RPG also follows the same approach for what shall be included in CM, like assumptions, algorithms; and what the elements shall be, like entities, actions, tasks, processes.

The study defines a process for M&S development, on which V&V activities and relations between those activities are depicted. RPG places CM development between planning and design activities, and emphasizes its importance. CM is defined as "collection of information that describes the developer's concept about the simulation and its constituent parts, and serves as a bridge between developer and user [11]". Remarkably, design is defined as "a translation of the information captured in CM to support their implementation in software and hardware [11]". In this way, RPG supports the usage of CM in design activities. RPG also supports the usage of UML in CM documentation.

VV&A RPG states that user, project management, developer, VV&A agent and SME shall all be included in CM development and validation. A remarkable argument suggested by RPG is that, CM artifact in a simulation corresponds to sketches and floor plans as construction artifacts. This argument makes meaning of CM clearer, and emphasizes that CM is essential in simulation development.

VV&A RPG defines a very similar development process for CM with Pace's. Differently, the study emphasizes the importance of usage of "common semantics and syntax", "common format DBMS", and "Data Interchange Formats (DIF) [23]".

2.2.4 CMMS / FDMS

CMMS is an effort conducted by DMSO to describe a set of methodologies and tools to develop mission space conceptual models. The program is initiated and directed by DoD M&S Master Plan in October 1995. Objective 1 of the plan is to "provide a common technical framework for M&S"; and includes three sub-objectives, "establishing a common HLA", "developing CMMS", and "establishing data standards [28]". They aim to increase reusability and interoperability of different simulation systems.

DoD has stated two serious problems in development process; for which conceptual modeling will have an outstanding effect to be solved. First problem is, "different simulation developers often rely on different sources for the same information"; and second is, "the information is not necessarily maintained for use in future simulations [29]". DoD proposes the solution to these problems as to develop CMMS for every mission area of DoD, and to develop necessary methodologies and tools for this aim.

Conceptual models of MS (CMMS) are "simulation implementation-independent functional descriptions of the real world processes, entities and environment associated with a particular set of missions [30]". DMSO designs CMMS "to serve as a first abstraction of the real world and as a frame of reference for simulation development [29]". As Sheehan clarifies, in simulation development process, CMMS places CM development between "End User Models" (which correspond to user requirements) and "Synthetic Representations" (which correspond to design) [30]. MS model in between two of them includes processes, entities, relationships and interactions included in MS as implementation independent descriptions. This approach is similar with Pace and VV&A RPG approaches.

The current situation of CMMS is totally ambiguous. CMMS was introduced in year 1995. At about year 2000, the study has been renamed as Functional Descriptions of the Mission Space (FDMS). Two years after that, at about 2002, FDMS also disappeared and at the same year, Knowledge Integration Resource Center (KIRC) emerged that seemed to be continuation of CMMS. Currently, formal documents on

DMSO site are available neither for CMMS nor for FDMS; the reason for changing from CMMS to FDMS; canceling FDMS, if so; or the situation and level of accomplishment of the study are not explained. As Lundgren states, “there seem to be no way of telling where the original CMMS stands today, how different approaches connect to each other and where the studies are heading [31]”.

As described in CMMS Technical Framework, the components of CMMS objective are as follows [7];

- **CMMS models** of real world military operations,
- **Common DBMS** to store and manage model libraries,
- **Technical framework** including technical standards, administrative procedures, and system infrastructure,
  - **CSS** for describing mission space,
  - **Process** for creating and maintaining CM’s,
  - **Data interchange standards** (DIF) for integration and interoperability.

CMMS describes standards for representing elements of simulation. CMMS defines “EATI (entities, actions, tasks, interactions) representation to define a CSS template [7]” which are independent of the environment used to capture CM. The elements to be utilized in this representation are listed as follows.
“Entity, state, role, event, verb, action, actor, entrance criteria, exit criteria, task, interaction, mission, mission space“

CMMS Technical Framework also defines a process to develop CM, that is composed of the following activities [7];

- **Develop Focused Context:** Focused context is the combination of MS and SS information. It defines the scope, resolution and fidelity for CM and development activities. Johnson specifies that a tool called MRM (Mission Requirements Module) was developed to determine missions, operations, tasks and relationships between them [29].

- **Gather Information:** The activity involves determination of scope and priority of information gathering activities, using mission partitions determined, and assigning organizations to gather types of information. Johnson again states that a tool named ADS (Authoritative Data Resources) Database was developed "that points to sources of data, their sources of authority, and information about quality engineering procedures [29]". ADS includes various topics like doctrine and operations, environment, equipment, force description, standards etc.

  In this step, CMMS aims to collect information from various sources in a set of Order of Battle (OB) DB. DIF's are defined to convert data from available DB's, and the tool checks the consistency and completeness of data coming from DB's.
- **Formalize Input Resources:** This activity involves converting the information gathered from authoritative resources into information of interest to developers. Information is organized into structured text descriptions with supporting diagrams. Templates and style guides are used to gather information from UML-style specifications and use cases for a variety of CASE tools.
- **Construct CMMS:** In this step, CMs are constructed as entity-based abstractions. CSS is used "to develop common understanding and to reduce mis-communication without restricting the language of warfighter or domain expert [29]". CSS includes "common dictionaries, common representation templates and tool specific guides [30]".

Many CASE tools can be used to construct CMMS. CMMS library tools convert them automatically to CMMS library by means of DIF.

To sum up, CMMS claims to define many procedures, standards and tools to develop CM, but solid results of the study are not available. Elements to be represented are listed, but it is not well defined to determine and depict them. UML and use-case like diagrams are suggested, but how to implement them are not provided. CMMS was a big potential to be a standard application in conceptual modeling area, revealing many details and suggesting comprehensive solution with standards, procedure descriptions, tools, utilities; but the project didn't go forward before being mature enough.

2.2.5 YEROOS

YEROOS is a project developed with cooperation from universities in Belgium, Switzerland, Argentina and Senegal. It was initiated in 1993, and stands for "Yet another project on Evaluation and Research on Object-Oriented Strategies". Project focuses on "object-oriented conceptual modeling for complex application domains, and methods for analysis and design of information-system applications [20]".

YEROOS defines conceptual modeling as "the activity of building a model of an application domain in terms of concepts that are familiar to actors in the domain in the real world"; in other words, "creating abstract representations of some aspects of physical and social systems and their environment in the world around us [20]".

YEROOS claims that conceptual modeling can be used in many areas of society, like management, manufacturing, cognitive science, information studies. Possible uses of CM are identified as to increase understandability of existing and new systems; act as a communication vehicle between modelers, system developers, specialists and end users; specify a documentation of functions and record of abstraction for system. It also states CM to be developed in early stages of system development, and to be input to design and implementation phases.

The study mentions that notations, tools and techniques are required to represent information. Thus, YEROOS aims to identify abstract relationships and integrate them into existing object-oriented models and languages like C++, Logtalk. The group states "they have conducted a number of case studies on conceptual modeling in various application domains (social security databases, automation of library management, and management of international banking credit) [20]". However, the details of their methodologies and applications are not provided.

2.2.6 FEDEP

FEDEP stands for HLA Federation Development and Execution Process. It is a high level framework defined to build HLA federations. HLA Object Model Template working group within US DoD started the discussions and published the first release

of FEDEP in 1996. Next five releases were sponsored by DMSO and supported by SISO; and version 1.5 was released at 1999. Later, at 2000, FEDEP was published as a recommended practice by IEEE, after other HLA documents. The last version of IEEE Recommended Practice is published as 1516.3 HLA FEDEP at 2003.
HLA is a standard developed by US DoD and used to develop distributed interactive simulations that connect sub-systems (federations) with a common interface.

FEDEP is a process model to develop HLA federations that "defines generic, common sense system engineering methodology for HLA federations", and "it is not intended to replace existing management and engineering processes of HLA user organizations [32]". HLA FEDEP is an "iterative waterfall process" which is similar to software development process definitions and "it aims to guide simulation developers through federation development [33]". The number of steps is updated through versions, although the activities are similar. As of 2003, the steps are listed as follows in IEEE Standard [34].

- Step 1: Define federation objectives
- Step 2: Perform conceptual analysis
- Step 3: Design federation
- Step 4: Develop federation
- Step 5: Plan, integrate, and test federation
- Step 6: Execute federation and prepare outputs
- Step 7: Analyze data and evaluate results

Step 1 includes the activities of identification of user needs and development of federation objectives. The inputs for the step are "overall plans, existing domain descriptions, and available resources"; and outputs are "initial planning documents, and federation objectives statement" which will become input to next step [32].

The step of conceptual analysis is the most important activity for this study. Purpose of this step is "to develop an appropriate representation of the real world domain that applies to federation problem space and to develop federation scenario [34]". Also, federation objectives are transformed into high-level federation requirements. The

sub-activities of this step are, development of scenario, development of federation conceptual model (FCM), and development of federation requirements.

Similar to approaches of Pace and CMMS, FCM is defined as implementation-independent representation, and is a vehicle to transform objectives into functional and behavioral descriptions. Approach of FEDEP for requirement terms are slightly different from CMMS; "federation objectives" meaning "requirements", and "federation requirements" meaning "specifications".

The importance of FCM to be a link between objectives and design and development is emphasized in FEDEP. FCM is defined as "a description of the entities and actions that need to be included in federation in order to achieve all federation objectives [35]". The approach is again similar to CMMS, as FCM includes determination of entities, actions and relationships. Moving further, although not specifying a notation to be used, FEDEP implies to depict static and dynamic perspectives in FCM. Static relationships can be shown with any kind of association like generalizations or aggregations. To depict dynamic relations between entities, sequence of actions and triggers can be used. Attributes and parameters, if possible, can also be identified. FCM shall also include assumptions and limitations for model. FEDEP states the selection of technique and format for FCM development and documentation to be recommended, but in any way, does not provide any options or guidelines on how to develop the technique.

The inputs of the FCM development activity are federation objectives statement coming from Step 1, federation scenario coming from previous activity in current step, and authoritative domain information and existing CM's if available. FCM will be used as a direct input in next activity in this step, development of federation requirements, together with federation objectives and scenario. Federation requirements are explained to be "based on federation objectives, testable", and "provide implementation level guidance needed to design and develop federation [34]". Considering this definition, federation requirements seem to be detailed and close to design. Every entity in FCM shall be documented in requirements.

FCM is used as inputs to latter steps, starting from federation design. It is exploited to select federates and prepare federation designs. Also in federation development phase, FCM is used to develop FOM and establish federation agreements.

Looking in detail to the steps, CM stands in the middle of defining federation objectives (or user requirements) and federation requirements (or specifications). In the higher level, CM is between defining federation objectives, and designing and development of federation. From this situation, we understand that FEDEP sees CM as both a means of defining federation requirements and an input to design phase as the structural basis for design and development activities.

2.2.7 SEDEP of Euclid RTP 11.13

Euclid stands for "European Co-operation for the Long-term in Defence". It is a collaboration of many institutions from European countries to study on common defense problems. There are "Research and Technology Programmes (RTP)" conducted within Euclid. RTP 11.13, named as SEDEP, consortium "comprises of 22 European companies from 13 nations, and with a budget in excess of €17 million [36]". The programme started in November 2000.

SEDEP stands for "Synthetic Environment Development and Exploitation Process". Its major initiative is "to promote the use of Synthetic Environments (SE) in Europe [37]". SEDEP does not only define a process to develop SE, but also aims to provide tool suite and an integrated development environment. By means of this framework, SE can be used in different domains, SE systems can be developed more effectively, and reusability of existing products increases.

SEDEP is developed based on FEDEP. SE is used as HLA term "federation". As SEDEP documentation mentions, "major difference between FEDEP and SEDEP is that SEDEP is more than just a development process since it highlights use of SE to support lifecycle of higher-level processes [38]". SEDEP provides more guidance by providing best practices, checklists of things to do and a framework including tools to support every level. SE of SEDEP may contain and link simulation systems, model, real-life equipments and people as one system.

Main artifact of SEDEP is prototype SE Development Environment (SEDE). SEDE is comprised of the following parts [37];

- **SE Process (SEDEP):** Definition of activities to create and implement SE.
- **SE Management Tool (SEMT):** Management and configuration control of SE data.
- **SE Tools:** Support for various activities defined by SE process.
- **Repository:** Facility for exchanging data between SE tools and for storing data.
- **SE Knowledge Base:** Information for users about available SE assets and best practices on developing SE

SEDEP includes 8 main steps, each of which includes sub-activities and products. The steps and the associated products are listed as follows [38].

- **Step 0 - Analyze user needs:** User Needs Analysis Documents
- **Step 1 - Define federation user requirements:** User Requirements Documents
- **Step 2 - Define federation system requirements:** System Requirements Documents
- **Step 3 - Design Federation:** Design Documents
- **Step 4 - Implement Federation:** Federation Components
- **Step 5 - Integrate & Test Federation:** Federation Ready for Operation
- **Step 6 - Operate Federation:** Federation Execution Data
- **Step 7 - Perform Evaluation:** Evaluation Results

The purpose of Step 0 is to "define and analyze user needs in order to understand what results SE should provide [37]". It is like a preliminary work for start of the project. Developers decide how to use SEDEP in this step. User needs, project plans and objective are outputs. Also, all relevant data gathered about domain are outputs of this step. In Step 1, federation user requirements, scenario and evaluation objectives are described.

The purpose of Step 2 is "to define specification for a federation that will satisfy user requirements [38]". First, by using user requirements defined in previous step,

implementation-independent CM of the federation is developed. Then, by using CM and user requirements, system requirements are developed. Lastly, evaluation criteria to specify how to evaluate system are developed.

Focusing on CM, three parts are defined for a CM in SEDEP, “simulation concept, simulation elements, simulation relation”. Pace’s definition for simulation concept is followed in SEDEP. For simulation entity, SEDEP has the same EATI approach of CMMS. Simulation relations are specified as high-level interfaces between elements.

Although it is stated that FCM can be defined in many ways, SEDEP clearly supports the usage of UML to increase reuse. As Lemmers states, “standard XML Metadata Interchange (XMI) for UML is used to store and interchange an FCM [39]”. SEDEP defines two profiling mechanisms to customize UML. First one is “specific stereotypes” defined for CM, and the second is definition of “well-formedness rules” that define relations between SE elements and relations [39]. SEelement, SEdata, and SErelation classes are created within first profile, and the second one defines rules among them, like communications of entities.

SEelement classes can communicate with SEdata by means of other SEelement classes. SEelements can have “generalization (is-a) and aggregation (part-whole) relations [39]” with each other. Attributes and behaviors can be defined for them. SErelation specifies data exchange between SEelements. So, when there is SErelation, “SEdata, SEdataProvider and SEdataConsumer” shall be defined to identify interface [40].

As a result, class-like diagrams depicting inheritances between SE elements, and diagrams showing relationships among those elements are developed as CMs. CM is developed using “Conceptual Model Tool (CMT)”, which utilizes “Rational Rose and DOORS” or others, owing to “wrapper” architecture [39].

In the next phase, Step 3, high level and detailed design are developed. This step and the next step are realized by using “Federation Composition Tool (FCT)” that uses “FCM, scenario and requirements” as inputs [41]. On FCT, federation scenario,

system requirements and FCM are shown in a tree-like hierarchical view. In the same view, federates developed until then are also shown. For a CM entity, UML like diagrams developed are displayed. When a federate is selected, related information of that federate about CM, scenario and system requirements are shown to the user. By this way, these three artifacts are linked to design step.

The outputs of federation design activity are "Federation Design Specification, selection of federates, FOM document, and federation agreements [41]". By using the tool, user can document capabilities for matching CM and requirements to design, and trace design back to requirements and CM. Output is added to repository in a special format as XML DTD (design document type definition).

2.2.8 DCMF

DCMF is a study of Swedish Defence Research Agency (FOI) that began in 2003. DCMF stands for "Defence Conceptual Modeling Framework" and it is originated from CMMS. It is a framework for "making conceptual descriptions and models of military operations [42]", and consists of a set of tools, methods and techniques for knowledge capture, analysis, modeling, representation and usage.

Although they base their studies on CMMS, developers of DCMF claim that CMMS was "vague and unfinished". A lot of the necessary components, methodologies and tools were missing. DCMF consists of the following components [42].

- **DCMF Process:** It describes the activities to organize raw unstructured data into structured information. It details conceptual modeling process starting from knowledge acquisition, representation, modeling and up to usage activities.
- **KM3:** It stands for "knowledge meta meta model". It includes a tool to model acquired knowledge, by means of which final CM's are created.
- **DCMF Ontologies:** It is the knowledge base component of framework and provides semantics for CM. Military Specific Ontology (MiSO) design methodology is defined to design and develop DCMF-O.

Knowledge acquisition phase as the first phase has three activities, “determine focused context where the scope and delimitations of knowledge requirements are decided”, “identify authorized knowledge sources”, and “apply actual engineering (acquiring, gathering and documentation) of data [43]”. Linguistic, phonetic, morphological, syntactic and semantic analyses are conducted in this phase. In knowledge representation phase, syntactic and semantic representation and modeling of raw data obtained in previous step are conducted using tools like 5W (who, what, when, why and where), KM3. Also, “the information is mapped to a suitable ontology [43]” in this step. Knowledge modeling aims to transform the models to a more reusable format and merge with previously created models in knowledge repository, so that models are machine-readable. All of these activities are supported by tools. UML can be used as a modeling tool to confirm to semi-formal ontology type. It is then easily transformed to formal ontologies.

DCMF defines ontology as the specification of conceptualization; and basically aim of ontology is to enable knowledge sharing and communication. Ontology of DCMF is developed within MiSO methodology, which includes multi-layered architecture. Upper-ontology layer is used “to tie down the domain oriented concepts into more abstract real world concepts like entity, time, space [42]”. Middle layer is more specialized for military operations and modeling domain. The lowest layer may include many scenario/application specific concepts.

The upper ontology used for conceptual modeling is entity-based, defined entity in top level, then physical entities, objects and processes. In middle layer, ontologies like weapons, vehicles, terrorist and defense organization are developed.

To develop a CM from a scenario, DCMF first parses the natural language text using methods of 5Ws, ending up in raw text. By using available tools, related ontology is populated for the scenario, which is a semi-formal ontology. Again using tools, the scenario is then “transformed into structured, machine readable data in OWL (Web Ontology Language).

2.2.9 JSIMS (Joint Simulation System)

JCMMS (joint CMMS) which is a set of CM's developed by JSIMS (Joint Simulation System) is "perhaps the largest set of domain-oriented CM's ever developed [24]" and it is developed in coordination with CMMS. At the higher level, it was mainly developed as part of U.S. Army's Warfighter Simulation 2000 (WARSIM) program. CM's were used to develop Army's Functional Description of the Battlespace (FDB) to make information available for reuse by other simulation programs within program, especially in requirements analysis and design activities.

JSIMS aims to provide a synthetic environment to provide realistic, joint and highly comprehensive military training. JSIMS connects to "both C4I functions and field equipment [44]" and supports mission rehearsal and training activities.

Formalized Data Products (FDP) are used to capture MS information in JSIMS. FDP's are "structured MS Word Tables filled with natural language information [29]". CMMS FDP converter developed in MS Visual Basic converts data to CMMS format and integrates into CMMS library by using DIF.

Object oriented approach for developing CM was proposed by development team; but it was rejected because of the concerns that knowledge acquisition personnel would have difficulty in capturing information in true format. JSIMS CM includes the following [44];

- Process for planning, communicating, executing, monitoring and assessing military operations
- Organizations of services from weapon teams through major commands
- Equipment used to sense, destroy, defend, communicate and supply, including weapon systems
- Information items and flows between commanders, subordinates, liaisons, and intelligence agencies
- Descriptions of common battlespace interactions; sensors and signatures, weapons and damage, communications and logistics

Risner and Porter describe V&V process of JSIMS using CM; first, CM's are validated against real world; second, JSIMS software verifies with the descriptions contained in JCMMS [45].

2.2.10 JWARS

JWARS (Joint Warfare System) is developed by US DoD. It aims "to develop a closed-form simulation of joint warfare that represents joint functions, processes and component warfare operations [46]". JWARS includes representation of "C4, ISR, Information Warfare (IW), joint theater operations (land, sea, air), firepower, protection, logistics, MOOTW (military operations other than war), special operations, strategic and tactical mobility, and weapons of mass destruction [47]".

JWARS software development process did not include a formal CM artifact. Because of the need for VV&A activities, V&V agents decided to create a virtual CM from development artifacts between user requirements gathering phase and the design phase. For JWARS, that included "Joint Application Design (JAD) packages, CMMS document and algorithm descriptions [46]".

JAD's were derived from each JWARS process and they were requirements packages. They provided details about processes, entities, actions, tasks and interactions of JWARS processes. JWARS CMMS includes information of "entities, actions, tasks and interactions in the form of JWARS Enterprise Model of a joint task force [46]". Algorithm descriptions include each JWARS algorithm and associated conditions, limitations and assumptions.

JWARS CMMS was represented using simple flow charts. CMMS is developed in UML using StateMate CASE tool [29]. Finished parts are loaded to Databank and converted to CMMS library.

2.2.11 OOS CML

OOS CML stands for OneSAF (One Semi Automated Forces) Objective System Conceptual Modeling Language. This is an ongoing conceptual modeling effort

sponsored by USA Department of Army Deputy Chief of Staff. This group aims “to develop verified ISR specific CM’s to be stored in Defense Intelligence M&S Resource Repository (DIMSRR). They have developed a color-coded CM language (shortly OOS CML) and developed some models using the language.

The structure of OOS CML is a graphical description including links to related documents. Abstract model is based on JSIMS abstract model. The model elements defined in abstract model and relations are as follows [48];
“Category, characteristic, event (activates behavior), element (belongs to category, has characteristic, generates event), behavior (alters the state of element), modifier (modifies behavior, create and delete element), gamespace, battlespace, piece (located in gamespace), time”

Piece may be players (like tanks) and markers (like tactical positions). Gamespace may be an environment as physical battlespace or zone as abstract spaces. Events may be “physical event (collisions), information exchange (order, report), element coordination, decision, state change, simulation control, game state update and external events (outside of simulation) [48]”. It is remarkable that simulation control events are placed together with events about elements. Behaviors may be immediate (with no time) or cumulative (having duration). Modification that a behavior does may be canceling or overwriting behavior’s changing of element’s state, delay, change data or logic.

The group has planned modeling of various national, joint and future force sensors like SAR (synthetic aperture radar) of UAV (Unmaned air vehicle), electro optics for aerial common sensor and such.

OOS CML implements flow diagrams by using the abstract model elements and relations as described above. Flow diagrams are a good way to show how sensors or other ISR elements work. But the flow diagrams alone are not enough to carry all information related to the system. For example, there is no diagram showing the parts of an element; all information is embedded in the flow diagrams. This increases complexity of the diagrams and prevents the existence of separate repositories for

different aims. Although there is a well-defined abstract model of the language, the author of this study thinks that the example diagrams lack understandability and readability.

2.2.12 CM by Reverse Engineering in Aegis Tech.

A CM was developed for an implemented project in the leadership of Jack Borah. The simulation was a "HWIL (Hardware In the Loop) simulation using portions of real-world missile defense systems to test and to produce data to identify interoperability capabilities and limitations of system [49]".

As the project was already implemented and requirements engineering wasn't applied, the group applied reverse engineering on the implemented software. Although at the beginning, the team was thinking of employing a CM schema, later it turned out that it is appropriate for the nature of the project to document CM in text supported with diagrams.

In CM text, first, CM description is given, including CM parts and points of contact. Then, Pace approach is followed, and simulation context, simulation concept and simulation components are identified. Simulation context is expressed by providing purpose and intended use and the domain background. Simulation concept is expressed by providing description of natural environment and simulation executive.

Simulation is divided into components and for each component, where applicable, description of the component, component composition, possible states, tasks/actions/behaviors, relationships/interactions, events and data are provided. As it was observed that text was not enough alone to describe simulation components, UML diagrams are decided to be incorporated. First, static diagrams were constructed that depicts the elements in the system. To model the dynamism of simulation, sequence and activity diagrams are constructed that shows the interactions during simulation. As a result, although the team experienced the difficulties of developing a CM by reverse engineering, they started to observe that CM would be helpful for the rest of life cycle of the project.

Aegis has also suggested tools to support federation design in an HLA system. As Hunt states, the study emphasizes that "little has been written detailing how CMMS models would be used in designing an HLA federation [17]". Their study attempts to detail how automated tools can be used to implement CMMS to support HLA federation design and development. For this process, three frameworks of DMSO are utilized; HLA, CMMS and Data Standards. First, a CM development tool is suggested to develop CM diagrams using UML symbols and importing EATI of CMMS. Then, scenario-generation tool is suggested to provide "simulation-independent environment for deploying instances of objects in CM [17]". In the next step, federation composer tool is designed to make a comparison of HLA object models and conceptual object models. At the last step, execution planner tool is suggested to output FOM objects that the federates are responsible for.

2.2.13 CM & Data Repositories by BAE Systems

Atherton explains in his paper CM and data repositories developed at BAE Systems at Warton in UK, for their simulation systems including synthetic environments of air systems, by means of in-house tools they have developed [50].

The company applies a FEDEP based development cycle. They have divided CM tool development facility into two, one is dataset repository which is responsible for holding all CM data and on which operations like search are conducted. The other is CM tool by means of which CM's are developed and added to repository.

They have CMMS like approach for development of CM, as they state that CM should include "attributes, behaviors, interactions, relationships, events, tasks, actions, states [50]" of an entity. Additionally, they suggest addition of confidence level to parameters; and any project restrictions on the parameter should be specified.

In Conceptual Modeling Tool, each entity is a separate model. A set of these models are linked together with interactions and relationships to create a master CM. Drop down lists are used to create an entity; first, entity type is selected (such as sensor), then a real world entity is selected (such as SAR radar) and version of modeled entity is stated.

2.3 KAMA-C4ISR

This is the methodology developed within Turkish Armed Forces M&S Master Plan. Name of the project is C4ISR-KAMA, which aims to "develop a tool for CM development that provides a common modeling approach and a common repository for related personnel of simulation systems [51]", especially for C4ISR systems. By means of this tool, it is aimed that simulation developers in Turkey will follow a common approach, and a repository will be formed in time, that new projects may benefit and reuse CM information from previous projects. In this way, both time and effort will be saved and efficiency of M&S activities will be increased by using a common language among simulation personnel.

KAMA development is leaded by project group from MODSIMMER; M&S Center within METU. Components of the project are "mission space CM, common repository, CM development tool [51]" including CSS rules. While the tool provides an environment to develop CM, the repository keeps and integrates different CM's, analyses the models for integrity and conformance to rules, manages access and usage by users in different physical places, and in this way, provides reuse of CM's.

CM Development Tool includes four main components; "model development, data entry, reporting and repository access [51]". Model development component provides an integrated development environment for models in an easy way. Different perspectives of the same model may be viewed and models are represented in a clear way by means of CM standards defined in the project.

Data entry component saves the model elements to the system; in the mean time, verifies the model for semantics rules previously defined. By means of reporting, the user can output data in different levels. The access of users is managed by repository access tool.

KAMA divides CM development process into two main steps; which are, defining CM elements and basic relations, and later forming detailed definitions of model elements and relations. In more detail, the steps include the following activities [52];

- Collect authoritative information on CM
  - Define simulation objectives
  - Define simulation context
  - Define sources of authoritative information
- Identify model elements
  - Define high level assumptions and constraints
  - Define missions and generalization/specialization relations among missions
  - Define tasks for each mission and task (dynamic view)
  - Define actors responsible for missions (dynamic view)
  - Define relations between actors and tasks (roles) (dynamic view)
  - Define entities related with tasks (static view)
  - Define relations between entities (static view)
  - Define inputs and outputs for tasks (static view)
  - Define objectives and measures for objectives
- Develop CM diagrams
  - Develop diagrams as listed below
- Restructure CM
  - Define states of the entities
  - Define events between entities
  - Relate missions to command hierarchy
  - Define mission-entity cross relations
  - Define mission-state cross relations
  - Define high level algorithms to identify interactions among entities
- Verify and validate CM
  - Verify CM with respect to semantics rules
  - Verify CM with respect to syntax rules
  - Validate CM
  - Report V&V results

It is clear that to be able to develop diagrams, model elements need to be defined first. But during diagram development, new elements need to be added. So, an iterative approach between development steps is crucial and beneficial. There is not a predefined order of developing diagrams; but examining the diagrams, one can find

priorities between diagrams. Organization structure diagram cannot be developed before command hierarchy. Similarly, work flow cannot be developed before mission space; entity state cannot be developed before entity ontology diagrams.

As Eryılmaz, Bilgen and Molyer discusses, for semantic V&V of CM, some rules need to be defined [16]. An example of this is, every task shall be performed by an actor. Validation for conformance to some of such rules is done during model development; and for the rest, V&V analysis is to be conducted after model development.

As CM language, KAMA suggests a "meta-model based on MDA (Model Driven Architecture) and developed consistent with MOF (Meta Object Facility)", which is based on UML and includes all meta-model definitions [53]. KAMA suggests that, rather than using directly a language like UML which is specific to object modeling, a specific language including simulation domain concepts will enhance the efficiency of modeling. This does not mean KAMA directly exploits UML knowledge; rather it defines a profile mechanism to extend UML according KAMA needs.

Various methods are suggested to develop formal CM's until now. These include "process flow diagrams, activity cycle diagrams, petri nets, event graphs, UML, object models", "simulation activity diagrams, tables describing rationale and content and DEVS [5], [21]". As Brade suggests, usage of a formal model has many advantages including providing "unambiguousness to the model, efficient implementation of solution, platform independent specification and enabling automated V&V [55]". Automated V&V activities may include syntax and semantics checking, control and data flow analysis and testing. On the other hand, again as Brade discusses, by implementing formal models, "intuitive comprehensiveness and unlimited expressiveness is lost [55]". Zeigler suggests the use of informal and formal description of a model together. According to him, informal model "plays a fundamental role to stakeholders to grasp the basic outlines of the model and to visualize it within the framework of their prior conceptions about how things work"; and the formal model is the "unambiguous description of the model structure [56]".

Tolk emphasizes the importance of development of CM based on standardized methods. He thinks that "PIM's (project independent models) as defined in MDA are good candidates for this aim [19]"; and it should be possible to enhance software engineering standards such as UML to comply with CM requirements.

Usage of UML for conceptual modeling is suggested in many studies. Widely known approaches like CMMS, FEDEP, SEDEP, Pace all support the usage of UML, though not providing explanation on how to use. One of more detailed studies is Gustavson and Zimmerman's study [22]. They claim that use case and class views are enough to depict all information to be contained in CM; and suggest correspondence between CM and use case diagram elements. Additionally, they use class diagrams to depict attributes and behaviors of elements defined in use cases. In this way, all information they claim to exist in CM is depicted by UML diagrams. DMSO VV&A RPG also suggests use of use cases for formal CM; stating that use cases "can serve as the mechanism to move from requirements to design [11]".

An advantage of UML standard is, it allows coping with a dynamical description of a system. UML enables us "to describe, how a system and its components interact externally as well as internally [57]".

In KAMA, UML is used to formalize CM; but by defining a specialized modeling language that extends UML. A meta-model is defined that includes UML elements from which CM model elements are derived, attributes and capabilities of elements, relationship types between elements and rules for those relationships. So, we can say that, similar to KAMA, the methodology proposed in this study exploits well-developed UML infrastructure, but defines its own language to define CM.

While Karagöz discusses CM development process, he introduces two types of KAMA model diagrams to develop model in different perspectives [52]. Mainly, there exists static and dynamic perspectives; missions, tasks, actors and relation between actors and tasks are part of dynamic, and entities, relations between entities and inputs and outputs are part of static perspectives.

Two types of diagrams are introduced within KAMA, structural and behavioral. Structural diagrams include "structural entities in mission space and time-independent relations among them [54]". Entities are shown as UML classes. Behavioral diagrams depict time-dependent relations between model elements. A mission can be specified as task in another model, which increases reusability. State transitions are also shown in behavioral diagrams. Diagrams according to perspectives, and model elements used in diagrams are as follows [54];

- Structural
  - Entity Ontology – entity, inheritance, part-whole relation
  - Command hierarchy – entity, command unit, line relation
  - Organization structure – entity, actor, inheritance, role
  - Entity relationship – entity, inheritance, relation, part-whole relation
  - Mission space – mission, actor, objective, measure, responsible, realize, extend, include, achieve
- Behavioral
  - Work flow – task, flow, decision point, synchronization point, initial state, final state, actor, input-output, state, input relation, output relation
  - Entity state – state, event, transition, initial state, final state

We should note that, like most of the approaches previously explained, KAMA aims to develop mission space CM. The processes explained above describe activities for mission space CM; and mainly aims to utilize CM in requirements collection and analysis activities. It does not contain any information on simulation space. KAMA states that mission space CM is placed before development of simulation requirements, and simulation space CM is placed after development of simulation requirements and before design [16]. It is also stated that simulation space CM includes simulation entities corresponding to elements in mission space, simulation control capabilities, control of input and output to software, operating system and hardware infrastructure. KAMA also does not provide any explanations on how to use CM in design.

2.4 Discussion of Approaches and Applications

Considering information on various approaches and applications provided in previous sections, we observe that there are many studies on CM from important researchers and institutions. This proves that CM is seen as an essential part of simulation development; and in the future, to be able to develop higher quality simulations with less budget, CM needs to be developed.

First CM thoughts have emerged to aid in VV&A activities. The importance of Sargent and Davis is that, they initiated the thought of "implementation-independent model".

Pace's work is very important in literature, as it has been the basis for many latter approaches. His definitions of simulation context, simulation concept, mission space, simulation space, simulation elements have been used more or less the same in DoD VV&A RPG, CMMS, FEDEP and SEDEP, which are the major studies in the area. The EATI structure is also specified by Pace.

Pace also outlined basic properties of a CM and how CM development process should be like. Considering many definitions and determinations he has made, he accelerated the development of CM, but he has not suggested any formal methodology on how to determine elements, develop and document CM. He suggested "scientific paper" approach to document CM, which does not formalize CM development process.

CMMS had been a great expectation to formalize CM development process, provide standards, procedures and toolsets. Basing the study on Pace's definitions, CMMS provided a detailed process description and started formalization by defining concepts like CSS, DIF, diagramming. Although the study disappeared, the concepts it defined are still utilized in the area.

Although having similar approaches to conceptual modeling, FEDEP and SEDEP differs from other approaches in that, they are not studies specialized on CM, but

they are frameworks to develop simulation applications. They specify the usage of CM in their process, and define input-output relations with other phases in the process. So they are helpful to place CM in higher level in SDLC. It is clearly emphasized in these approaches that user requirements are directly used as an input, and CM is used to develop specifications and design.

Pace discusses that there is a direct relationship between FEDEP products and Pace's definition of CM components. Federation requirements and federation scenario together are equivalent to "simulation context" of Pace, as "they establish constraints for specific application of distributed simulation [12]". FCM is equivalent to simulation concept, and simulation elements correspond to federates of the federation.

Importance of FEDEP is that, it is suggested as a recommended practice by IEEE as a standard [34]. FEDEP suggests the determination of technique and format for CM, and suggests the depiction of elements and relations on diagrams, but does not suggest any specific techniques within the framework.

SEDEP introduces the concept of "synthetic environment". Together with detailed process definition and repository, SEDEP also suggests many tools and knowledge base to automate simulation development at all steps and increase reusability of products. SEDEP implements the usage of UML to develop CM, but customizes UML by profiling mechanisms and develops class-like diagrams. In this way, SEDEP specifies the development and form of CM, however one perspective is provided, which the author of this study thinks that it is not enough to show different static and dynamic relations. The usage of CM in design is provided by data interchange formats, although FOM document is developed as a direct output.

DCMF goes further in automating CM development by defining ontologies and automated processes to parse scenario texts. Usage of UML is suggested. But to parse CM, comprehensive scenario definition should be developed. Also, how to use CM in design is not clear.

To sum up, it is observed that there are not formal procedures and standards available for use to develop CM. Some approaches tried to develop such methodologies, but could not complete it. Some couldn't provide a complete solution to develop formal CM, for example the procedures were not clear enough for others to use, or they tried to make conceptual modeling of system in limited perspective.

Other problem in CM field is that, in almost all studies, CM is divided into MS and SS, and only the guidelines on how to develop MS CM is provided. Some studies give only a definition of SS and states that SS CM is also required for a complete CM, but none of them provides any detailed description on how to develop SS CM. Also, although most of the approaches emphasize the importance of CM as an input to design, they could not provide detailed prescriptions on how to use CM in design.

KAMA is a study that aims to fulfill the gaps in conceptual modeling field and to provide a complete set of procedures and tools to develop well-defined CMs. This study aims to move this objective one step further, and extend KAMA to also develop simulation space CM and to use CM in design activities.

CHAPTER 3

PROPOSED CM DEVELOPMENT METHODOLOGY

This chapter includes a complete and detailed description of the proposed methodology for CM development. The methodology proposed in this chapter is an extended KAMA methodology, and based on KAMA. Though KAMA provides a comprehensive solution to conceptual modeling that both grabs current knowledge in the field and guides the M&S personnel with a very well defined method (which is a well-known flaw of conceptual modeling at the moment); the author aims to further enhance the method by defining extensions to develop simulation space CM and to use CM in design, and in this way strengthening the place of CM in SDLC with clearer relations between requirements analysis and design phases. The studies presented in this and following chapters furthermore include knowledge from current approaches and applications and are supported with the author's own experiences and observations in M&S field.

Three types of CM development and documenting methodology can be evaluated. First one is scientific paper approach, which Pace suggests, describing CM in natural language like a scientific paper. Second approach is developing CM representations by exploiting design methodologies used in software development cycle. Pace's idea on this design formats is that, they may not be able to capture all aspects of CM [12]. Third approach is using ad hoc methods specific to each project developed.

As the aim of KAMA and this study is to develop a standard methodology for CM development, enable formal representations that automate development and usage of CM, and form a repository that will enable reuse; neither scientific paper nor ad hoc approach is appropriate. The aim is also not to employ whatever formalism used to describe simulation design, changing from project to project. The approach includes usage of formalisms developed for software design, but exploiting that information to develop related CM development standards and language. This is what KAMA has already implemented and the author of this study thinks as the best possible practice. So, a formal methodology is to be developed based on an existing design standard, UML; by redefining a language for conceptual modeling using UML infrastructure. KAMA has already developed a graphical modeling language to develop CM; and based this language on UML as the meta-language. In this way, it is aimed to ease the transition process from CM to software analysis and design steps, at which UML is already widely used. As stated, UML is just used as meta-language in this process, CM development methodology redefines its own elements, diagrams and rules; but depicts them by using UML notation. So, CM's developed using this methodology need to be developed using a CASE tool that implements UML. The same approach for conceptual modeling language is followed in this study for developed extensions, in this way, original KAMA and extensions are in harmony, and it is easy to implement new definitions to existing infrastructure.

3.1 CM Development Process

The very first thing to define how to develop CM is to define the steps of conceptual modeling process. As explained in ***2.3 KAMA-C4ISR***, KAMA defined CM development steps as collecting authoritative information, identifying model entities, developing CM diagrams, restructuring CM and verifying and validating CM. Keeping basic activities KAMA has defined, some steps are updated to enable the integration of extended issues in proposed methodology. The following steps of CM development process are identified; and the activities to be conducted are explained in detail within the context of this methodology in the following sections.

- Collect authoritative information
  - Define simulation objectives
  - Define simulation context
  - Determine sources of authoritative information
- Identify model elements
  - Define high level assumptions and constraints
  - Define model elements
  - Determine elements as MS or SS
- Develop Mission Space CM diagrams
  - Develop seven types of mission space CM diagrams
- Develop Simulation Space CM diagrams
  - Develop five types of simulation space CM diagrams
- Verify, Validate and Finalize CM
  - Verify CM with respect to syntax and semantics rules
  - Validate CM
  - Release version and update repository
- Develop high level design
  - Develop UML design diagrams by using CM

Requirements analysis and CM development activities are highly dependent on and affected from each other. Considering perspectives from outstanding approaches (as explained in ***2.2 History, Current Approaches***), the best approach to place CM in simulation development process is as follows; 1) Compose/obtain user requirements, 2) Develop CM by using user requirements as input, 3) Develop system requirements by using CM as input, 4) Develop design by using system requirements and CM as input. Lacy also states that as a result of DMSO experts meeting [24], it turned out that simulation experts think the best approach is to develop two different CM's, mission space CM as a result of which requirements will be developed; and simulation space CM as a result of which design will be developed.

On the other hand, Pace emphasizes the "non-orderedness" of requirements and CM development in the following sentence: "Simulation requirements and CM development each stimulate and derive from other; CM development may even begin

prior to completion of simulation requirements; and requirements may be updated while CM is developed [4]". KAMA also emphasizes that requirements analysis and CM development activities shall be iterative.

Considering these, CM development process may be initiated at any phase of requirements analysis activity; with the start of, during or after. Researchers agree that the early the CM development is started, the more its benefit to development. So the best practice will be starting conceptual modeling activities with start of simulation requirements development, developing both of them iteratively, and finalizing them together. Iterative development means, developers turn back and update requirements by using learnings from CM development; and vice versa, CM is updated by using learnings from requirements development. This continues through the whole CM development process, that is why "update requirements" activity is not placed in CM development process definition that is listed above.

By means of this iterative approach, error-free and complete requirements and CM are achieved. Also, flexibility is provided about the time conceptual modeling activity is to be started; and simulation developers benefit CM independent of different approaches they follow in development life cycle. The following sections, where the steps of conceptual modeling process are described, assume that a requirements analysis document (user or system) is available to use as input to CM, because it would not be possible to describe the process considering iterativeness between requirements and CM development.

3.2 CM Development Methodology

A methodology is developed by KAMA that defines elements to exist in CM, and diagrams to show different perspectives. As a result, a formal CM is developed. Similar to KAMA, the methodology proposed in this study exploits well-developed UML infrastructure, but defines its own language to define CM. The CM to be developed by the proposed methodology is implementation independent. However, it is not totally possible to talk about an "absolute" implementation-independence for

CM; if we want to include all aspects of simulation requirements into CM. Especially simulation space may include aspects dependent to implementation. DMSO VV&A RPG states that "simulation space component of a CM is seldom totally implementation independent [11]". Implementation independence is desirable, particularly during initial development of CM. However some dependencies are acceptable if they occur as a result of practical considerations based on requirements of intended application. Pace also thinks that "if the objectives of the federation indicate that a particular simulation is to be involved or that particular kinds of live forces are to be involved, then CM cannot be totally implementation independent [4]". He accepts that CM should have "reasonable" implementation independency, but a CM will typically have some level of implementation dependence to be fully responsive to simulation requirements.

The activities that will be explained in following sections define the steps of the proposed methodology for CM development.

3.3 Step 1 – Collect Authoritative Information

The first step of CM development includes activities of information collection to develop CM. The main input for this activity is requirements documents (user or system) finished before developing CM, or currently being developed. Simulation objectives define boundaries for simulation context. Simulation context, which is the information provided by the domain that simulation addresses, shall be examined carefully, related data shall be extracted and references shall be determined for CM development team.

There are many techniques defined to collect information and store it in a systematic way in knowledge engineering discipline. However, these methods are beyond the scope of this study. In the following sections, the activities to collect information are defined; but systematic of information collection, organization and storage are not mentioned.

3.3.1 Define Simulation Objectives

The simulation objectives are the first information that constraints what the simulation will do. They define the user expectations on what the simulation will do in general terms. Simulation objectives can be defined in user requirements document; but if not yet defined; it should be documented before starting CM development. The objectives define the domain of the simulation. Simulation context shall be defined and information shall be gathered according to objectives.

3.3.2 Define Simulation Context

According to Pace's definition, "simulation context provides authoritative information about the domain which the simulation is to address [12]". Examples of simulation context are laws of physics, coordinate systems, general doctrine and tactics, operational modes of devices. Simulation context can be thought as collection of information from and references to domain sources of the intended simulation. Simulation context establishes "the boundaries on how a developer can properly build or modify a simulation for intended use [11]".

Simulation context for the system to be developed shall be identified according to simulation objectives. First, domain(s) of the system is to be identified, going from general to more narrow as much as possible. Then relevant authoritative sources available for that domain shall be determined and examined, giving priority to the ones the user prefers. Necessary information from those sources shall be collected in a data repository within the project; keeping in mind this repository would enlarge widely in later phases of project.

It is also important to note that collection of authoritative information on simulation context is not one for all activity. Through the time CM is developed, simulation context definition will be extended and more information will be collected.

3.3.3 Determine Sources of Information

This activity in fact shall be carried on throughout information collection activity and rest of CM development. All sources of information shall be determined and references shall be kept. Also, whenever information is used, sources of that specific information shall also be kept as a reference.

3.4 Step 2 – Identify Model Elements

In this step, CM elements will be defined by using requirements document. In this study, element definitions of KAMA are utilized; but new element types are added and some elements are updated for extension objectives of this study. Set of elements to be identified are listed in section ***3.4.2***, with explanations of the elements.

3.4.1 Define high level assumptions and constraints

High level assumptions and constraints for CM and model elements are identified while defining elements, and this process will be continued in the next phase of CM diagram development. Other than high level assumptions and constraints, system wide assumptions and constraints that both belong to real world aspects and simulation application shall be identified and documented in this step.

3.4.2 Define model elements

In this step, the elements of CM are determined by using the requirements. Six rules of Pace (***2.2.2 Pace Approach***) (which are also accepted by DMSO) shall be kept in mind while defining the elements; 1) There should be a specific element for every item in requirements, 2) There should be a specific element for every item of potential assessment interest related to purpose of simulation, 3) There should be real world counterparts for every element as much as possible, 4) Elements should correspond to standard and widely accepted decomposition paradigms in problem domain, 5) Elements required for computational considerations should be used carefully, 6) There should be no extraneous elements. The model element types to be defined and attributes of each type of element are listed in below sections.

3.4.2.1 Entity

Entity is any being in mission space that has specific properties on its own. An entity may be an actor, organization, facility, network, material, equipment or object. Some types of entities will be differentiated as specific element definitions, like actor. The entity is to be shown by a "Class" in UML notation. The properties of entities are as follows;

Name: A specific name for the entity.

Code: A unique code for the entity. A systematic way to define a code should be specified, because this code should be specific among all entities in mission space.

Type: It shows the entity that this entity is derived from. This is the same inheritance relationship as object-oriented languages.

Kind: This specifies if the entity is facility, force, feature, software, information, material or equipment. The list may be extended. It is not mandatory to specify kind for parts of an entity, as it may be meaningless. For example, the part of an airborne force is wing, which does not have any capability on its own. If the type is a specific one like actor, then rather than indicating it here, the name "Entity" is changed. The explanations for the possible kinds of an entity are as follows.

Facility: Physical establishment used to conduct system-wide operations, like buildings, computer stations etc.

Information: Entity that does not have any functions or existence on its own, but yet is a collection of definitions; like database records.

Force: Entity that has different types of attack capabilities and intelligence on battlefield; like airborne vehicles, soldier, troop.

Equipment: Entity that is used for and has specific capabilities on battle field, which is managed by another entity, like electronic warfare equipments, radar etc.

Material: Entity that has specific capabilities on battle field, managed by another entity, used for once and consumed, like bullet.

Feature: Cultural, line or point entities on terrain, like bridge, tactical point etc.

Software: Any kind of software artifact.

Assumption: Informs any assumptions about the entity (for real life or simulation)

Constraint: Informs any constraints about the entity (for real life or simulation)

Attribute: Lists all the attributes of the entity. Attributes of an entity may belong to real life, or it may be a kind of attribute that is only specified for simulation. For example, maximum acceleration value of a mobile entity exists and may be measured in real life. But no default acceleration value is specified in real life for an entity; it is defined to determine behavior in simulation. Then, attributes shall be classified as simulation or real, to differentiate between mission space and simulation space. If possible for the moment, unit of the attribute shall also be specified. An initial value can be assigned to the attribute, which can be a real value or a symbolic value. If the attribute is an enumeration, possible list of values can be defined.

Attributes may be fixed, that is initiated from database value and does not change on run-time, like physical properties of entities. Other attributes may be initiated with an initial value (probably obtained from database) and calculated and updated during run-time, like location and speed of entities. The property of attribute to be fixed or variable shall be stated at CM.

Behavior: Lists all the behaviors and capabilities of the entity. Like attribute, an entity has some behaviors that come from its nature in real life; and some additional behaviors specified for simulation. So, behaviors shall also be specified as MS or SS.

State: The states that the entity may be in. These states and transitions between these states are shown in a separate diagram.

Relation: This part lists the relations of the entity between model elements in different diagrams. This need not be stated as a list; but rather shown on diagrams. (Modeling tool should automatically extract a list of relations)

3.4.2.2 Actor

Actor is an entity whose kind is “actor”. Actor is the entity that is responsible from executing tasks. An actor may have more than one role in mission space. Actor inherits all properties from a regular entity; but preferably, it is shown with a stick man symbol in diagram.

3.4.2.3 Role

Role is the definition of different tasks of an actor in different missions. The same actor may be involved in different roles in different missions. In this way, the actor definitions are added to system once that is less prone to change; while roles can be

dynamically created for different needs. Role “executes” or “is responsible for” different tasks. Role can also be shown with a stick man, but in a different way than an actor. Role has the same properties with an actor.

3.4.2.4 Object

The object is a specific entity that exists at real time, during scenario execution. In conceptual modeling, it is used to show possible specific types of entities. For example, the airborne vehicle force can have many objects, like F-16, Seahawk etc. The entity that the object belongs to is written on the name of the object. The object has the same properties with its entity, but behaviors of the object are not shown on diagram. The attributes are also need not to be shown; only the fixed attributes that are assigned a specific value (like a fixed value from database), or the variable attributes for which assigned value at a specific time at runtime are required to be displayed are shown with the values they are assigned. In this way, objects both serve as an environment to store the instances of entities required by user, and data related to those entities whenever obtained by developers. Usually, objects are created from entities that have behaviors; which are usually entities of kind force, material or equipment.

3.4.2.5 State

State is one of the possible conditions in which an entity may exist or in which the entity executes an operation at one point in time. The transition between the states of an entity is shown on diagrams. States are shown with the “state” notation of UML.

3.4.2.6 State Transition

State is a relation between states of an entity such that an entity moves from one state to other when certain conditions are satisfied. Transition is depicted with a regular transition relation on UML notation. The following information is provided to depict a transition:

Source entity of transition,
Result entity of transition,
Transition condition,
The event that causes the transition.

3.4.2.7 Event

An event is an occurrence that triggers transition between states. An event causing the transition shall be searched for whenever there is a state transition. The triggering event is depicted on the transition line between two states.

3.4.2.8 Mission

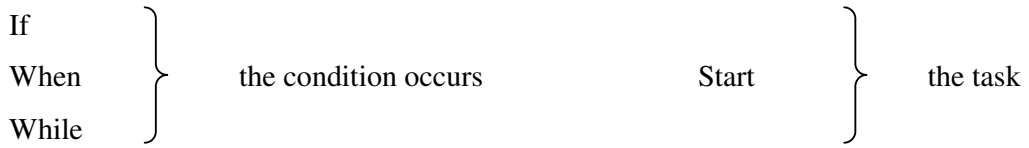
Missions are the high level tasks that the simulation system is expected to fulfill. The missions of the system may be thought as the objectives for which the system is built. It is similar to definition of task in EATI approach of CMMS. The mission involves collaboration of two or more entities and tasks between those entities. Mission is shown with "use case" notation in UML. The properties of mission are listed below.

Name: A specific name for the mission, expressed preferably as an order.

Code: A unique code for the mission. A systematic way to define a code should be specified, because this code should be specific among all missions in mission space.

Explanation: A detailed explanation for the mission is helpful.

Pre-Condition: The conditions that are necessary to execute this mission. Pre-conditions can include completion of a mission, conformance to a time condition, a change in location and such. In fact, the conditions are depicted as relationships in diagrams associated to other elements, but any additional conditions shall be added if necessary. If possible, the conditions shall be stated in following forms;



Post-Condition: The conditions that are necessary to be executed before the end of the mission, or the conditions that ends the missions when they occur. The end conditions are also depicted as relationships in diagrams, but any additional conditions shall be specified if necessary. Post-condition can also provide conditionals during the time the mission is executed. Two different end conditions shall be defined for successful and unsuccessful ending of the mission. If possible, the conditions shall be stated in following forms;

If / When / While } the condition occurs — Pause (pending) / Terminate (unsuccessful) / Finalize (successful) / Continue (loop) } the task

Assumption: Informs any assumptions about the mission (for real life or simulation)

Constraint: Informs any constraints about the mission (for real life or simulation)

Geographical properties: Determines the geographical conditions or locations on which the mission is conducted. The terrain entities defined in CM or references to external entities may be associated with a mission.

Input/output: They are depicted as separate elements on the diagrams.

Objectives: They are depicted as separate elements on diagrams.

Measures: They are depicted as separate elements on diagrams.

3.4.2.9 Task

Although task has the same properties with the mission, it is referred to as more detailed activities in order to satisfy a military objective. It is shown with the activity symbol in UML notation. Subtasks of a task can be defined by placing the subtasks in the boundaries of a general task notation. Subtasks can have relations between each other that normal tasks have.

3.4.2.10 Objective

This determines the objective that is to be expected to be satisfied by the mission or task. It is connected to measures in order to decide if the task has reached its objectives. It is shown with the class symbol in UML notation. It has the following properties.

Name: A name to determine the objective.

Explanation: The description of the objective.

Measure: Lists the measures to evaluate if the objective is achieved. They are shown separately in the diagram and connected to related objective.

Achievement status: Lists the conditions of measures under which the objective is successful or unsuccessful. These are stated as expressions like “if measure 1 is greater than 0”.

3.4.2.11 Measure

It is measurable information defined to decide if the objective is successful. It is shown with class symbol in UML notation. The properties are as follows.

Name: A name to determine the measure.

Unit: The unit of the measure.

Value: The value to evaluate the measure.

3.4.2.12 Input/Output

The inputs necessary to execute the task and the outputs resulting from the execution of the task are listed. They are work products. They are shown with the class symbol in UML notation. The properties are listed below.

Name: A name to determine the input/output.

Code: Short code to determine input/output. It should be unique in the mission space.

Explanation: Detailed explanation for the input/output.

Attributes: Lists all attributes of the input/output. It defines all components and types of it. Input/outputs may be derived from each other and can have part-whole relations.

Assumptions: All assumptions about input/output.

Constraints: All constraints about input/output.

Relations: This part lists the relations of the input/output between other input/outputs and missions and tasks. This need not be stated as a list; but rather shown on diagrams. (Modeling tool should automatically extract a list of relations).

3.4.2.13 Algorithm

The algorithm is also a type of entity that is used to determine the algorithms to be used in the system and the associated entities. In this level, it is not necessary to list all algorithms to be used in the system; the ones to be included in the model are especially the ones which are specifically mentioned by the user that are not left to the developer's initiative, or basic behaviors implemented by other entities in the system. It doesn't aim to provide any implementation knowledge. Algorithm entity is in fact a detailed explanation of a behavior in the system, which is used by one or

more entities. The algorithm itself does not have any attributes; it uses attributes of other entities as inputs. The attributes of the algorithms are listed below.

Name: A name to determine the algorithm.
Code: A short code to determine algorithm.
Explanation: The detailed explanation on the algorithm.
Inputs: The input attributes to be used in the algorithm. These are especially important to specify the fidelity of the algorithm. For example, it is important to mention that 5 DOF orientation data is enough for modeling of a missile.
Outputs: The parameters that are calculated by the algorithm. This output is used by the entity that uses this algorithm.
Assumption: Any assumptions to be applied for the algorithm. This decision will highly affect the implementation.
Constraint: Any constraints for the algorithm (like computational complexity)
Relation: The entity to which the algorithm is related. A "used by" relation is defined between entity and algorithm. If the algorithm uses another entity as input, an "input" relation may be defined between them. They are shown on the diagram.

3.4.2.14 Relationship and Relation End

The general relation defines the basic relation association between two model elements. In the below items, specific relation types will be listed. For relationships that do not fit in one of those types, this general relation may be used. A relation name may be provided on the relation line on diagram. A generic relation is shown as a straight line between elements. The following information may be given on relation ends.

Quantity: Indicates the number of possible entities in relation with the other element. This can be used if it is meaningful for the relation type used. Quantity is shown close to relation end as described in ***3.4.2.16 Part-whole Relationship (p/w)***.
Direction: Indicates the direction of communication between entities. An arrow is placed in the direction of communication at the relation end. If communication is two-ways, a straight line is used.

Role: A role for the relation can be defined on relation ends, to specify the kind of relation. Possible role types can be "contains, is contained in, uses, is used by, access". When a role is defined, relation name may be canceled.
Constraint: If necessary, a constraint can be specified for any of the relation types.

For specific relationship types that are explained below, abbreviations indicated next to relationship name on the heading will be used on diagrams, to increase readability.

3.4.2.15 Inheritance Relationship (inh)

It is a relation between a general and more specific entity. This relation indicates that similar entity in one side of the relation inherits attributes and behaviors from the other. The one with the general properties is the ancestor, and the more specific one is the sibling. The sibling inherits all attributes and behaviors of its ancestor. These properties may be used exactly the same by the sibling, or it may be updated. Additional attributes and behaviors may be defined for the sibling. An entity may not be its ancestor. This relation is valid for every type of entities including actors.

3.4.2.16 Part-whole Relationship (p/w)

This relation is used to show the components of an entity. An entity may not be its own part. A diamond-shape is placed at the relation end close to the "whole", and the other side shows the part. The quantity of possible parts of an element can be indicated in detail by depicting quantities on both sides of relation line, interpreted separately. Quantities can be one of the followings;

0..1 : Zero or one elements.
n..m: n to m elements.
0..* or * : no or any number of elements.
1 or n : exactly one or n elements.
1..* : any number of elements.

As examples, when quantity is placed as "1" for part, it means there can only be one from that part in a whole. If it is "0..*", there can be no or any number of that part in the whole. When quantity is placed as "1" for whole, there can be one whole that part can be in. When it is "1..*", there may be many wholes for the part.

A p/w relation can be a strong type, when a part can belong to just one whole. In strong p/w relation, quantity of whole can be 1, and quantity of parts cannot be range, but can be a static number like 1 or n. The diamond for strong relation is shown as black filled.

Together with the p/w relation, a role can be defined like in a generic relation, as described in ***3.4.2.14 Relationship and Relation End***. This provides more information on relation between the part and the whole.

3.4.2.17 Line Relationship (line)

It is used to define military command hierarchy among superiors and inferiors. An arrow is placed at the relation end close to the "superior", and the other side shows the inferior. An entity may not be its own superior.

3.4.2.18 Responsible Relationship (resp)

It is the relation between mission/task and the role, to show that the role is responsible for that task. An arrow is placed at the relation end close to mission/task.

3.4.2.19 Realize Relationship (rlz)

It is the relation between mission/task and the role or another entity, to show that the role is responsible to implement or realize that task, or the entity realizes that task. An arrow is placed at the relation end close to mission/task. A role may have both responsible and realize relations to a task. Generally, for missions and high level tasks, responsible relation; for low level tasks, realize relation is used. Each mission and task shall have a role that realizes or is responsible for that task.

3.4.2.20 Extend Relationship (ext)

It shows the extend relation between missions/tasks. It may be used to make a generalization as an abstract task (that is not implemented), but the specific tasks extended from that task are implemented. The arrow is placed close to high level mission on relation line.

3.4.2.21 Include Relationship (inc)

This type of relation is used to depict lower level missions under a high level mission. The arrow is placed close to detailed mission on relation line.

3.4.2.22 Achieve Relationship (achv)

It is the relation between the objective and the mission. It is used to show achievement relation for the objective. More than one objective may be defined for one mission.

3.4.2.23 Input Relationship (inp)

It is used to depict the relation between the inputs of a mission/task and the mission/task. More than one input may be defined for a mission/task. This relation can also be used for entities.

3.4.2.24 Output Relationship (outp)

It is used to depict the relation between the outputs of a mission/task and the mission/task. More than one output may be defined for a mission/task. This relation can also be used for entities.

3.4.2.25 Own Relationship (own)

It is used to determine the relation between an actor and a role. An actor may own more than one role; a role may have more than one actor.

3.4.2.26 Success Criteria Relationship (succ)

It is used to show the relation between an objective and the measures of that objective to evaluate if it is successful.

3.4.2.27 Used By Relationship (usdb)

It is used to determine the relationship between an algorithm and the entity. An algorithm may be used by more than one entity.

3.4.2.28 Control Flow

Control flow is used to depict the flow between the tasks. The flow is shown starting from one task to the other task, through which the arrow is placed. Control flow may be used between the tasks, control and synchronization points and initial and final states. A regular control flow between tasks A and B in order indicates that there is a sequential execution between two tasks, indicating Task B begins when Task A ends. The flow may also be disjoint, meaning Task B starts some time after Task A ends. If such a flow exists, it shall be stated on the flow as “disjoint”, and any other condition to start Task B shall also be stated (like time constraint).

3.4.2.29 Decision Point

It is used to depict conditional flows, conditional loops and selective situations in work flow. One or more tasks may enter decision point, and one or more tasks may be routed from decision point. All control flows entering and exiting decision point shall be named. Decision point is depicted with a diamond symbol as in UML notation.

The conditions that cannot be expressed on the diagram may be stated in "Pre-Condition" and "Post-Condition" properties of the mission, although depiction on diagram is preferable.

If, when and while conditions and the behavior on the task to be executed can be depicted on control flows exiting decision point. If, when and while condition is written closer to decision point side, and the resulting activity (Start, Pause, Continue, Terminate, Finalize) is written on the task side. Notice that terminate, pause, finalize and continue are applicable only if the task is already started before. The following explanation shall be kept in mind to determine the usage of them;

IF the predicate is true, behavior is this, else do another behavior.
WHEN means you will do this; however you don't do it until the predicate evaluates to true.
WHILE means as long as the predicate is true, do the activity, else do another activity.
When nothing is indicated, the default action is IF.

3.4.2.30 Synchronization Point

It is used to determine parallel (concurrent) execution of different tasks (fork) or to join tasks that are already executed in parallel (join). Synchronization point is shown with a join/fork symbol in UML notation.

The concurrent execution of tasks can be in two ways; begin-begin and end-end. When a synchronization point is put and tasks are placed in parallel under that, it means that it is a begin-begin execution; all the tasks start at the same time (but may finish at different times). When multiple tasks are placed in parallel and

synchronization point is put under these tasks, it means that is an end-end execution; all the tasks finish at the same time and the start times are arranged accordingly. To explicitly define concurrency type as the opposite of what is explained above, or in other cases like tasks are placed between two synchronization points; it should be written on synchronization point.

It is also possible to state not the concurrent beginning and ending times for tasks, but stating limits for start and end of tasks. To state that Task A begins before Task B (or any other tasks under synchronization point) begins, the flow shall be named as "begin priority". If it is required to state that Task A ends after Task B ends (or any other tasks under synchronization point), the flow shall be named as "end priority". If the synchronization is required to be stated for smaller group of tasks (like two), the synchronization points shall be grouped accordingly.

3.4.2.31 Initial State

It is used to indicate the initial state of the work flow. Initial state symbol in UML notation is used.

3.4.2.32 Final State

It is used to indicate the final state of the work flow. Final state symbol in UML notation is used. More than one final state may exist (like successful and unsuccessful). Different symbols shall be used for that.

3.4.2.33 Note

A note may be used for any model element, to provide additional information. A relation is placed between the note and the related model element. Note symbol in UML notation is used.

3.4.2.34 Package

Model elements may be grouped in packages, to increase understandability. Package symbol in UML notation is used.

The identification of elements activity does not have any order in itself, and it is an iterative process. To ease the process, it is best to define entities as soon as possible. There are not any guides provided for element identification in KAMA. The

developer is expected to utilize requirements document and extract elements by considering element definitions. In this study, a small guide is provided to extract elements from requirement statements. The guide does not aim to provide the developer with a complete list of elements; it just aims to be a starting point for element definition phase.

To start this activity, main assumption is that there is a systematically prepared requirement document available, whether user or system requirements; and objectives of the system are indicated in that document. Developing software requirements is a very wide area on its own, and there are various methods for this activity, which is not the topic of this study. Basically, as explained in IEEE standard, the requirements specification should be "complete, consistent, modifiable and traceable [58]". Also, all the requirements statements should be complete (must fully describe functionality), correct (must accurately describe functionality), feasible (possible to implement), necessary (customer really needs), prioritized, unambiguous and verifiable. Of course, it is possible to develop CM even the requirements specification isn't perfect; but the quality of the requirements will obviously affect the quality of the model.

At first level, main model elements shall be extracted from requirements. It should not be forgotten that this is a preparation step for developing diagrams, so all elements and all details may not be finished at this step. Important model elements in the following types are expected to be identified for the system;

Entity	Attribute	Task
Actor	Behavior	Objective
Role	Mission	Input/output

Although there is no way to write excellent requirements, if the requirement statements follow a technical style, it will be easier to extract the model elements. As suggested in the book Software Requirements by Wiegers, it is preferable that requirements statements are "short, have proper grammar, active voice is used and

ambiguous words are avoided [59]". Next step is the conversion of these natural language specifications to formal specifications.

There are some approaches to analyze natural language to develop representations like heuristic methods and formal analysis methods that aim automatic extraction of representations, as explained by Firat [21]. In this study, we propose some guidelines to extract elements from sentences; but they are not aimed to fully cover all possible situations, or be a method to automate the extraction process. The following method is rather a heuristic method to ease the extraction process.

A simple declarative sentence is used to form statement, which is the type of sentence used in requirements. A declarative sentence is composed of subject and verb, and changing from sentence to sentence, adjective, noun, pronoun, adverb, preposition. The elements in sentence structures may be used to extract related model elements as shown in ***Table 1***.

As explained in the table, many basic elements may be found out by examining the word structure. But this table only guides the developer to determine the elements; the developer still needs the related military and modeling knowledge. Moreover, there are many other elements to be extracted that cannot be found with this method, but depends on the skills of the developer. Examples are events, states, most of the objectives and measures, assumptions, and constraints. The following are some examples to determine model elements mentioned in ***Table 1***.

Req.1 The system shall be composed of off-line and run-time interfaces.

- Here "The System" is the subject of the sentence. It is an entity and it indicates the system that implements the simulation the user is aiming.
- "Off-line interface" and "run-time interface" are nouns of the sentence. They mention two new entities for the system.
- "Compose" is the verb of the sentence, but it does not mention a task on its own. Considering the preposition "of", we understand that there is a "part-whole relation" between the subject and noun of the sentence.

Table 1: Matching of Sentence Elements to Model Elements

Sentence Element	Model Element	Description
Subject	Entity	Subject usually specifies an entity, or provides more information for an entity. The entity may also be an actor or a role.
Noun	Entity	A noun usually represents an entity. It may also be an actor or a role. If a mission or task is indicated in the sentence, the nouns may be input, output, objective or measure.
Personal pronoun	Actor, Role	When there is a personal pronoun (in the form of I, me, my or mine), there is an actor or a role in the sentence.
Adjective	Attribute of the entity	The adjective usually determines an attribute for the noun it is modifying
Adverb	Information on relation or attribute	When the adverb is used to modify verb, it may provide information on a relation. When it modifies an adjective, it may provide information on entity's attributes.
(Transitive) verb	Task / Mission	As the transitive verb is an action verb that requires a direct object, which states a task. This is the type of verb that is frequently used in requirements. If the subject is an actor, the verb may mention the role of the actor, or the task that the actor is responsible for. If the verb is not a general action about the system; the verb may indicate a behavior for the entity it mentions.
Preposition	Relationship	Together with verb, it may define a relation between the task and the entity.

Req.2 Tactical manager shall manage the run-time interface.

- "Tactical manager" is the subject of the sentence and a new entity. As "tactical manager" is a personal pronoun that refers to a person, it is an actor in the system.
- "Run-time interface" is the subject of the sentence, and an entity that is previously defined.
- "Manage" is the verb of the sentence. It describes a role for the tactical manager, also a task he is responsible for. So, "run-time manager" is a role of "tactical manager."

Req. 3 The run-time interface shall simulate airborne vehicles.

- "The run-time interface" is the entity that we have previously defined.
- "Airborne vehicles" is the noun of the sentence. It is an entity that will be "simulated" in the system.
- "Simulate" is the verb of the sentence, and as a transitive verb, it is meaningful with its noun. "Simulate airborne vehicles" is a task of the "run-time interface" entity.

Req.4 Maximum one hundred airborne vehicles shall be simulated by the system.

- "Maximum one hundred airborne vehicles" is the subject of the sentence. "Airborne vehicles" is an entity that we have defined before. "one hundred" is the adjective that modifies this entity, so it describes an attribute of the entity; which is "number of vehicles". "Maximum" is the adverb for the adjective "one hundred", it modifies the adjective. Considering these, "maximum number" is an attribute for "airborne vehicle" entity and its value is "100".
- "System" is the noun; and implicitly it mentions the run-time interface (as we know that airborne vehicles are simulated by run-time interface). There is no new element.
- "Simulate" is the verb and does not mention a new element.

The model elements extracted from four requirements are as follows;

Entity [Name: **System**]

Entity [Name: **Off-line Interface**; Relation: **Part of System**]

Entity [Name: **Run-time Interface**; Relation: **Part of System**]

Actor [Name: **Tactical Manager**]

Role [Name: **Run-time Manager**; Relation: **Owned by Tactical Manager**]

Entity [Name: **Airborne Vehicle**; Attributes: **Max.Number (100)**]

Mission [Name: **Simulate Airborne Vehicles**; Relation: **Realized by Run-time Interface**]

3.4.3 Determine elements as MS or SS

In the previous step, all types of model elements are determined. Some of those elements belong to mission space (MS), while some of them belong to simulation space (SS). As we develop two separate models for MS and SS, we need to identify which elements belong to which space.

MS elements are the ones that belong to military operations domain. These elements exist in real life with determined properties and have nothing to do with the simulation system. The elements that exist in software or that owe their existence to simulation system are SS elements. Hardware parts of the simulation system, software modules, interfaces, network parts and software are all parts of SS.

Some elements are MS elements that exist in real life; but for them to be able to exist also in simulation system, some new properties need to be defined. In this situation, we name the element as an MS element and model it in MS, but add some properties that belong to SS. Those properties are shown as a different set on the MS diagrams, and they belong to SS model. All elements defined in previous section (***3.4.2 Define model elements***) are MS elements, but most of them may also be used in SS, or SS properties may be added to them. An entity may be an SS element on its own (like a software module), or SS properties may be added (e.g. adding a new attribute and behavior to an airborne vehicle to define it on HLA). An actor (and likely a role) may also be an SS element, when that actor has only roles for execution of simulation. Missions and tasks are MS elements, but inputs and outputs, pre and post conditions for them may be SS elements. Additionally, all relation types can be used in SS.

Identification of model elements and determination of these elements as mission and simulation space elements are explained here as two different steps, to be able to explain methodology in detail. In practice, it will be easier to differentiate MS and SS elements while determining those elements.

3.5 Step 3 – Develop Mission Space CM Diagrams

Being identified the model elements and specified them as mission or simulation space, now the next step is developing mission space CM diagrams, which is the most important step of conceptual modeling. In this step, only model elements that belong to mission space are used for developing diagrams, although those elements may have properties that belong to simulation space. Very few exceptions may exist; for example, if a simulation space element is directly in relation with MS elements, it can be placed in mission space diagrams.

As known, although proposed CM methodology is based on UML, it defines its own model element types. Likewise, new types of diagrams to be used in CM are determined. Many researchers mention that one perspective is not enough to model a system. We see that UML also follows this idea and presents many types of diagrams of different perspectives for the user. Although many current CM approaches proposes the development of one kind of model (like CMMS, SEDEP), we follow the same approach also for conceptual modeling that, to compose a complete model, more than one perspective is required. In the following section, seven different types of diagrams depicting different perspectives of CM will be explained.

During diagram development process, iterations will be required to turn back and define new elements and add them to previously developed diagrams. This need arises because the developer can make better decisions on how to define model elements and place them in diagrams as he reaches later phases of model development and as he observes relations while developing diagrams. Also, during diagram development, the developer will realize that there are some elements in the system that are not mentioned explicitly in the requirements. Moreover,

supplementary elements may be required just to depict intended situation in diagrams. For all these situations, model development process shall be iterated several times and new elements shall be added until the developer is convinced that CM thoroughly represents what the user wants, is close to real world, is design independent and has no extraneous elements.

The same iterative process is also required for diagram development. There is not a predefined order to follow while developing CM diagrams. Modeling is not a one step activity, and the order of diagrams to develop is not strict. That is, after finishing one type of diagram and developing the second type, developer will probably need to turn back and make some additions to diagrams of first type. This will go on through the whole model development cycle.

3.5.1 Entity Ontology (EO) Diagram

This type of diagram is used to depict entities in the system with details. All generic entities of the system (excluding specific ones like actors and algorithms) and related objects are placed, but not all relations between entities are shown in EO diagrams. Rather, this is a specialized diagram for inheritance and part/whole relations between entities. All properties of entities, like attributes, behaviors, assumptions and constraints are specified. To sum up, in EO diagrams, the entities and objects that have inheritance and part/whole relations between each other are shown with their properties, and those relations between them are depicted. This diagram shows static relations between model elements, so this is a structural diagram. Once this diagram is defined, information it carries can be utilized in other diagrams. However, it is not expected to be finalized in one step but rather iteratively updated.

All types of entities that exist in the system are introduced in these diagrams. These entities may include platforms and players that exist in the system and various player systems including weapons, sensors, and countermeasures. These may be defined in a detail level required by the system. Other than these entities, one special aspect that can be modeled using Entity Ontology diagrams is the environment. There are studies introducing approaches to develop CM of environment in simulation systems. For example, Dobey suggests a five-step process for developing an environmental

representation that extends from FEDEP. He defines environmental concept model as "an implementation independent, unified description of the synthetic natural environment for a simulation application [60]". In our study, environment is proposed to be modeled by using EO diagrams. Environment may be depicted as a separate entity in the system, which is composed of parts like meteorological and oceanographic. Terrain is also a part of environment. These parts may also have sub-parts and they may have specific attributes. Like all other entities in the system, relations between these environment entities and the properties they have are shown in EO diagrams. But the usage of environment entities is not restricted to this diagram. The entities that are described in EO diagrams can be used in other diagrams to show relations of environment entities with other entities in the system.

3.5.2 Command Hierarchy (CH) Diagram

Command hierarchy diagram shows the actors and line relationship between those actors. All the actors that have roles in missions and tasks are depicted on this diagram. This diagram is also used as a basis to develop organization structure diagram. This diagram shows static relations between model elements, so this is a structural diagram. This diagram type is not expected to change frequently in DB, as the information is static and many projects may use it in the same way once defined.

3.5.3 Organization Structure (OS) Diagram

Organization structure diagram is used to depict actors and roles that take part in a mission or task, and to depict relations between them. Actors may be selected from ones defined in CH diagram, or can be derived from them. Hence, entity types of actor and role, and own relation between actor and role to show the roles that an actor has are used in this type of diagram. This diagram also shows static relations between elements. As the same type of actor may have many roles in different systems, this diagram will need to be updated more frequently.

3.5.4 Entity Relationships (ER) Diagram

In this diagram, relations between entities other than the ones shown in EO diagrams are modeled. These may be entities that are not general, but specifically associated

with the missions modeled in MS diagrams; or the ones about roles conducted for specific missions. Input/outputs, algorithms and relations of them are also shown in this diagram. The developer may define new relationship types, if the predefined types do not fit developer's need. Although it is preferred to define all inheritance and part/whole relations in EO diagrams, if required, these relations may also be used in this diagram type. This is again a structural diagram as the relations are static.

3.5.5 Entity State (ES) Diagram

This diagram type is used to depict the states of the entities and the relation between them. Transition relations are defined between those states, and the events causing those transitions are also shown. Initial and final states are also depicted to show transitions between them. Conditions for the transitions need to be specified on the relation line to clearly model the entity states.

An important note for this diagram is that, there may be more than one set of states for an entity, for different situations. If so, different ES diagrams shall be formed for each set of states. In this diagram, dynamic relations between events are shown, that depend on events and conditions. So, this is a behavioral diagram.

3.5.6 Mission Space (MisSp) Diagram

Mission space diagram informs the users about what the system is expected to do. Missions are actually high level tasks, but they are more generalized and inform about the objectives of the system. Missions shall be in harmony with and shall describe simulation objectives. Many elements may exist in MisSp diagram, including missions at different levels, objectives and measures of those missions, extend and include relations between missions, actors and roles executing missions, inputs/outputs and relations between them. In this way, mission space of a simulation system is depicted with a MisSp diagram. As static relations between missions and related entities are shown in this diagram type, it is a structural diagram.

3.5.7 Work Flow (WF) Diagram

This diagram is used to show all kinds of detailed tasks in the system in a dynamic manner, similar to activity diagrams in UML. The tasks are mainly smaller parts of missions. WF diagrams include flow of tasks, describing operations and activities including previously defined entities. WF diagrams are the most detailed and "crowded" diagrams. Almost all model elements may be used in WF diagrams to describe how a task is executed, flow of tasks in time, effects of different conditions, synchronizations and decisions and how a task is started and finished. Because of prolixity of this diagram, decisions on how to cut down tasks and how to define detail levels are important.

To sum up, all diagram types, and the model elements and relations that can be used in each diagram type are shown in ***Table 2***.

Table 2: Summary of MS Diagram Types

Diagram Type	Elements	Relations	Type
Entity ontology	Entity	Inheritance, Part/whole	Structural
Command hierarchy	Actor	Line	Structural
Organization structure	Actor, Role	Own	Structural
Entity Relationships	Entity, Input/Output, Algorithm	Part/whole, input, output, used by	Structural
Entity State	Entity, State, Event, Initial State, Final State	Transition	Behavioral
Mission Space	Mission, Actor, Role, Input/Output, Objective, Measure, Other Entities	Responsible, extend, include, achieve	Structural
Work Flow	Task, Actor, Role, Decision Point, Synchronization Point, Initial State, Final State, Role, Input/Output, State, Objective, Measure, Other Entities	Control flow, input, output, responsible, realize, include, achieve	Behavioral

Another important point is that, to increase reusability and readability of diagrams, it is best to model diagrams in a hierarchical manner starting from general and going to detailed. Depending on the capabilities of modeling tool, links can be placed on higher level diagrams, which open more detailed models for some part of high level diagram when required. For example, in mission space diagrams, similar missions can be grouped under a name, and detailed missions can be shown as connected to that general mission. In this situation, for modeling considerations, general mission may even be an abstract mission that in fact does not exist in real life.

While completing the development of diagrams, it is also important to add supplementary information to the diagrams. Information to be added to each diagram can be listed as follows;

- Version number
- Publish date
- Revision history
- Author(s)
- Related project(s)
- Sources of information (if possible, separately for each element)

Diagram development process for MS can be finalized by adding this information to each diagram before adding them to common warehouse.

3.6 Step 4 – Develop Simulation Space CM Diagrams

Next step of modeling is developing SS diagrams, which describe SS properties, like how the simulation system will work, main structure; including both hardware and software components. SS is an essential part of CM with MS, as CMMS Technical Framework states and names as “Conceptual Model of the User Space (CMUS) [7]”. Because of its structure, SS component of a CM cannot be totally implementation independent. However, these implementation independent elements are usually the ones that the user wants to specify as a requirement. For example, although implementation issues, users usually want to specify how to give inputs to system

(joystick, keyboard, touch screen), how many stations will exist and the organization and hierarchy of stations. Although not restricted to the list, main issues to be considered in SS models can be provided as follows;

- Structure of the facilities in simulation system environment
- Structure of software applications, modules and states
- Roles in simulation system to start, stop, pause system, to manage system, to define DB records and others
- Inputs and outputs
- Runtime controls of simulation
- Input types for simulation
- Network hardware and software

As SS diagrams provide information for a specific system, it is usually not directly reusable by other systems. Still, SS models may be used by other systems to get a perspective of what elements may exist in SS, how the system works and to evaluate the reusability of system. In this way, developers may find the chance to complete requirements regarding to SS in early phases of development.

Although there are some studies in the literature to develop MS conceptual models, there are almost no studies on how to model SS. In some studies, SS is named as "user space". To model SS, we will follow the same approach explained for MS model development. Model elements that are specified as "SS elements" will be used in modeling. Exceptionally, a few MS elements may exist in SS diagrams, if that element has relationships with SS elements. Like MS, it is important to develop different types of diagrams to represent different perspectives of SS. So, most of the diagram types that are explained in section ***3.5 Step 3 – Develop Mission Space CM Diagrams*** will be used to model SS. In following sections, an explanation of how these diagrams will be used (or not used) for SS modeling will be provided. Also, a guideline on how to model some widely known characteristics of SS as listed above is provided.

Just like the situation in MS modeling, the need will arise to add new elements to diagrams and update diagrams. The developers shall iterate over SS diagrams until they agree that SS diagrams are complete, representing all aspects of SS and what the user wants about conducting of simulation system activities. Also, developing diagrams in a hierarchical manner will again increase readability and reusability.

3.6.1 Entity Ontology (EO) Diagram

Like in MS modeling, this diagram is used to show entities, and inheritance and part/whole relations between those entities. Many types of entities may exist in SS, like hardware (work stations etc.), software and network. Inheritance and part/whole relations in SS elements are shown using this diagram, in the same way as MS diagrams.

For SS, this diagram will especially be used to introduce simulation system facilities, hardware, software and network and modules that compose these entities. Looking at this diagram, one can grasp a view of physical structure of the modeled simulation system. Although there is no constraint, entities in this diagram will probably have no attributes or behaviors, as they represent high level components of system.

3.6.2 Command Hierarchy (CH) Diagram

This type of diagram is used to show actors in the system and line relations between them. Actors may have roles specific to simulation system, but actors themselves are real world entities. This means that they are MS elements; therefore this diagram is not meaningful for SS modeling.

3.6.3 Organization Structure (OS) Diagram

This diagram is used to show SS roles of MS actors and relations between them. This diagram is an exception to SS diagrams, as actors are placed in the diagrams as MS elements. This is necessary to be able to show actors that implement SS roles.

This diagram type may be used to depict roles that conduct simulation system activities; like management of facilities, decisions of trainings to be conducted,

starting of executable software, entering records in DB's, selection and start of scenarios, runtime controls etc.

3.6.4 Entity State (ES) Diagram

SS entities may also have many states, and this diagram is used to show the relations between these states. This diagram is especially important to understand how simulation system works. A clear example is running modes for software. Usually, simulation system has modes for different activities in system (like offline and runtime) that cannot work at the same time. This diagram is used to show such simulation system states and transition between them.

3.6.5 Entity Relationships (ER) Diagram

This type of diagram is used to depict all kinds of relations between entities, other than the ones shown in EO diagrams. Because of its wide scope, this diagram is very useful for SS modeling. Other than existing relation types, different types of relations may be required in SS that can range widely from system to system. This diagram type is especially important to meet variable structure of SS elements.

For SS, this diagram may especially be used to depict relation between software and hardware modules, like which software module runs on which hardware; and relation between states of the system and software modules. For example, the relation between most of the software entities is shown in work flow diagrams, especially the ones which are controlled by the actors. But the possible relations between entities which are controlled by the system, not the actors, are not shown. Such a relation can be depicted on ER diagrams.

3.6.6 Mission Space (MisSp) Diagram

Mission space diagram provides information about what the system is expected to do, that is the objectives of the system. SS does not deal with the objectives of the system, but rather deals with operational structure of the system. So, as the name indicates, mission space diagrams are not used in SS conceptual modeling.

3.6.7 Work Flow (WF) Diagram

This diagram may be used to show many tasks executed in simulation system as part of SS. Mainly, this diagram is used to state what activities are conducted by actors to operate simulation and how they are conducted; it doesn't contain inner facilities of system. Examples of issues that can be shown in WF diagrams are provided in following list. However the usage of this diagram is not restricted with this list, it can be used to model other system properties, depending on the simulation system.

- The activities needed to start the simulation system (shown with tasks and control flows)
- The activities conducted by the actors to operate the system and execute different facilities in the system, that are specified in other diagrams (task, control flow, role, realize relation)
- Different situations that occur during execution of tasks, decisions, conditionals, parallel executions, loops (decision point, synchronization point, control flow, conditionals depicted on decision point and control flow)
- The inputs and outputs of different activities and input/output relations (tasks, control flow, input/output, input and output relation)
- Which actors and roles execute which tasks (task, role, realize relation)
- How definitions in the system are defined in the system, like defining the properties of a player (task, control flow, decision and synchronization point)
- Runtime controls for scenario (like start, stop, pause) and other elements in scenario (like controlling and killing players, changing environment conditions), or other controls in system (like replay controls) (different controls can be depicted as tasks that can be executed under different conditions)
- How user inputs are entered to system (user input type can be depicted as a constraint on realization relation)

For each of the tasks conducted, the software entity on which this task is conducted can be specified on task notation. The software entities are the ones specified on SS EO diagrams having kind "software". The exact code of entity as specified in SS EO diagram shall be used. Although main objective of WF diagram is to specify

activities conducted by actors, in this way, the software entities that serve these activities are explicitly identified on this diagram. To summarize, the diagram types that can be used in SS conceptual modeling, with the elements and relations that can be used are shown in ***Table 3*** below.

Just as MS diagrams, to finalize SS diagrams some supplementary information shall be added. This information is the same as the ones listed for MS diagrams, version number, publish date, revision history, authors, related projects and sources of information. Diagram development process for SS can be finalized by adding this information to each diagram before adding them to common warehouse.

Table 3: Summary of SS Diagram Types

Diagram Type	Elements	Relations	Type
Entity ontology	Entity	Inheritance, Part/whole	Structural
Organization structure	Role, MS Actor	Own	Structural
Entity Relationships	Entity, Input/Output	Part/whole, input, output, customized relation	Structural
Entity State	Entity, State, Event	Transition, initial state, final state	Behavioral
Work Flow	Task, MS Actor, Role, Decision Point, Synchronization Point, Initial State, Final State, Role, Input/Output, State, Objective, Measure, Other Entities	Control flow, input, output, responsible, realize, include, achieve	Behavioral

3.7 Step 6 – Verify, Validate and Finalize CM

3.7.1 Verify CM with respect to S&S rules

Verification of CM has two aspects, verification with respect to syntax and semantics. Syntactic validation is mostly applied during model development, by

means of development environment KAMA aims to provide. Development tool specifies the kind of elements, what information to enter for the elements, and only permits the relations and properties of elements that are defined in methodology. In this way, wrong model structure is prevented during development.

For semantic validation, some rules are defined in meta-model level, which are again controlled during model development. The rest of the verification activities are to be applied after model development. When CM is finished, KAMA aims to apply an overall analysis for the syntactic structure by means of automated tools. In this way, model is verified syntactically and semantically.

By means of the rules defined in meta-model and post-development analysis activities, the conformance of model to templates, existence of related elements with their properties, relations between elements, cross relations between different types of elements and unit consistency issues are all tracked and secured.

3.7.2 Validate CM

CM validation is about the level CM meets simulation objectives. It is hard to establish standard techniques to be used for validation. The most important technique is reviewing by third party experts. Mostly used reviewing techniques are mental running of the model to check for feasibility; interaction analysis to check consistencies between different diagrams and interactions; analysis of functions, inputs and outputs. KAMA development environment aims to provide an easy interface for subject matter expert to conduct these activities.

3.7.3 Release version and update repository

Before completing CM, diagram identification (code, version, date) and diagram change history shall be added to all diagrams. Also, purpose of CM and reference to requirements documentation shall be provided to complete CM of a system. In this way, CM is easily specified and understood by possible users.

When CM is verified and validated, it becomes a reusable artifact, not for only the developers of that project, but also related personnel of many other M&S projects. To increase the reusability, completed CM's need to be stored in a systematic way. So, the completed CM is uploaded to common repository and tagged with a version. From that time on, CM is available for usage of M&S personnel.

By means of the completed CM, the requirements can also be validated; because CM includes a complete set of domain concepts. Also, a simulation system which is built on a validated CM will have easier V&V activities and will have less defects and deficiencies in latter phases.

3.8 Step 7 – Develop High Level Design

Most of the approaches explained in previous chapters (***2.2 History, Current Approaches***) state that CM shall be used as a direct input of design. DMSO VV&A RPG states that simulation concept (which includes mission space and simulation space in DMSO terminology) "serves as the mechanism by which M&S requirements of an intended application are transformed into detailed simulation specifications and then into an associated simulation design [11]".

Although almost all approaches discussed until now emphasize that CM is an input to design activities; there are not many studies suggesting a methodology on how to use CM for design. A rear example is a study conducted in Aegis Research Corporation; that uses CM to automate the development of FOM for HLA federates [17]. Although this is a good example that develops a solid artifact using CM as a direct input, the output covers only HLA design of the system.

This study aims to provide guidelines on how to utilize CM in design in a wider perspective. The most contemporary approach in developing M&S projects is using object-oriented design and development methodologies. Zeigler states that "model building and simulation execution is made easier by means of technological advances like object-oriented programming [56]". Considering this fact, this study also

assumes the design and development of related projects to be conducted using OOD paradigms and UML methodology. In fact, conceptual modeling language suggested in this study has also an object-oriented approach, organized around "elements" rather than "activities". This makes transition from CM to design easier.

In the latest UML 2.1 Superstructure standard published in February 2007, total of 13 diagram types are specified to be developed for system design [61]. Similar to diagrams specified in conceptual modeling, diagrams are organized as structure and behavior diagrams, according to dependence on time. Some of these design diagrams may be developed as a skeleton and a high level design can be formed by using available mission space and simulation space CM diagrams. It will then be easier for the developer to insert detailed design decisions to system by using the high level design. In this way, the developer may skip a few steps of design. In following sections, each UML 2.1 diagram will be handled separately and transformation from CM diagrams to that UML diagram will be discussed. The resultant design artifact will be referred as "high level design".

3.8.1 Class Diagrams and Package Diagrams

Class diagrams show the main parts of the system. It is a static view of the model, listing the attributes and behaviors of the entities and main relations between objects.

A class can be defined as an element that defines the attributes and behaviors that an object is able to generate. Classes correspond to entities that are depicted in "Entity Ontology Diagrams" in mission space CM.

The entities in simulation space CM EO diagrams cannot be used as direct classes, because they explain the structure of simulation system, not the domain. But they are helpful in class diagrams in other way. Package diagrams are used to organize related elements into headings, and used mostly for class diagrams. The entities of SS EO diagrams which have "software" as "kind" attribute can be used to form packages that include the specified classes, and other packages that the developer will add new classes. The entities defined in MS EO diagrams mostly specify classes that exist in run-time software module of simulation systems. Other software entities in SS EO

diagrams specify other packages, like network software, DB edit, scenario management etc. As these are implementation-dependent issues, the details of them are not specified during conceptual modeling activities; however the main issues are specified as separate entities in SS conceptual modeling activities. Capturing a view of main packages he should develop, the developer decides the classes belonging to packages identified in high level design.

In a class definition, the attributes and behaviors of that class are placed. The attributes and behaviors of entities specified in CM are exactly placed in class diagrams. Both MS and SS attributes and behaviors are used in class diagrams. Additionally, for necessary attributes, "Set" and "Get" methods shall be defined. Although units are not placed in class diagrams, they should be placed in design documents and code as specified in CM. If an initial value is stated, it is also placed in class diagram. Also, the information that the value of attribute is fixed or variable shall be used in design. The scope of attributes and behaviors in CM are not defined, as it is a design decision. It should be defined during design.

The developers shall review these attributes and behaviors in class diagrams at later stages of design. There will be a need to add some new behaviors to execute detailed algorithms of the system, but in general, for a well-built CM, the defined behaviors should meet all possible behaviors the system is expected to execute.

The relations shown are generalization, aggregation, association, composition or usage. The generalization relation is used to show inheritance. It is shown as a line with an arrow in the side of parent. This relation type is shown as the "inheritance" relation in EO diagrams of CM. Just like as described in ***3.4.2.15 Inheritance Relationship (inh)***, the attributes and behaviors of parent element are inherited by child elements.

The aggregation relation shows that an element contains or is composed of other elements. This relation type is shown as "part/whole" relation in EO diagrams. The quantity is transformed to multiplicity, the strong p/w relation is transformed to "composition" relation, and the roles defined are used in the same way.

Association relation implies two elements have some relationship. It is used to depict generic relations used in CM. The direction, multiplicity, name and role (if they exist) are used in the same way. The association relation usually implies an implementation of instance variable in one class. The developer shall specify such instantiation in later design phases according to information provided on relation.

For any relation type, when a constraint is defined, it should be placed also on related class diagram. Also, the roles defined on relation ends shall be used in the same way.

Another element of class diagram is interface class. It is a specification of behavior, and all classes using it guarantee to support that behavior. Similarly, algorithm element in CM explains a behavior that is implemented by one or more entities in system. Therefore, algorithms may be translated to interface classes in high level design. Both algorithms and interfaces do not have any attributes on their own. An algorithm does not have complete entity properties on its own; it needs to be used by an entity like a behavior. Similarly an interface cannot be instantiated; only a class can implement it by placing operations. The relation "used by" between algorithm and entity is transformed to a link between interface and class in class diagram. Other relations of algorithm element (like "input") or related entities of input attributes (that exist in algorithm definition) are transformed into associations in class diagram with all the information, as explained in above paragraphs.

Table 4 below depicts the elements in CM diagrams and corresponding design elements, to summarize how to use CM to create class diagrams.

Considering ***Table 4***, it is observed that many of the class diagram elements can be formed using CM knowledge. Of course, it does not mean that this high level diagram involves all design decisions required for a complete class diagram. The developers shall investigate for appropriateness of existing elements and extra classes, relations, attributes, methods, etc. to mature the design. But it is obvious that the initial class diagram developed using CM is a good starting point for the design activities of the developer that reflects validated system decisions.

Table 4: CM elements used to develop Class Diagram

CM Diagram	Element/Relation	Class Diagram Element
MS EO	Entity	Class
MS EO	MS Entity Attribute	Class attributes - create set and get methods for them in later stages
MS EO	SS Entity Attribute	Class attributes - create set and get methods for them in later stages
MS EO	attribute units	- use in design documents
MS EO	attribute initial value	attribute initial value
MS EO	attribute value fixed or variable	- use in design documents
-	-	scope of attributes and methods
MS EO	MS Entity Behavior	Class methods – detail these methods in later stages
MS EO	SS Entity Behavior	Class methods – detail these methods in later stages
MS EO	inheritance	Generalization
MS EO	part/whole	aggregation
MS EO	strong part/whole	composition
MS EO & ER	quantity of p/w and other relations	multiplicity
MS ER	generic relations	associations carrying all information provided on CM relation – define instance variables in later stages
MS EO & ER	constraints on all relations	constraints on relations
MS EO & ER	roles on all relations	roles on relations
SS EO	Entities with kind "software"	packages that cover classes defined in first-level design
SS EO	other entities with kind "software"	empty packages for other components of simulation system - fill them with classes in later design steps
MS ER	Algorithm	Interface class
MS ER	"used by" between algorithm and entity	link between interface and class
MS ER	"input" of algorithm	associations

3.8.2 Object Diagrams

In UML, object diagrams are a special case of class diagrams. They don't provide any new architectural information, but they increase understandability of class diagrams and give information on run-time behaviors of elements. The attributes and behaviors of classes are not shown for objects, but only the attributes whose values are required to be shown are displayed. In this way, information on object at any point in run-time is provided.

In conceptual modeling, objects are created in EO diagrams for a similar objective, to give run-time information on entities. For this aim, object names of different types of entities are provided; if possible, specific values of fixed attributes are given, and mostly a generic value for run-time variable attributes is provided. These are used in the same way for object diagrams, and the run-time variables are updated as required in design.

The following table depicts the elements in CM diagrams and corresponding design elements, to summarize how to use CM to create object diagrams.

As explained, object diagrams do not provide much information about the architecture of the system, but create run-time understanding of system and provide storage environment for different object names and attribute values.

Table 5: CM elements used to develop Object Diagram

CM Diagram	Element/Relation	Object Diagram Element
MS EO	Object	Object
MS EO	Associations	Same associations
MS EO	Values assigned to fixed attributes	Values of attributes
MS EO	Generic values assigned to variable attributes	- update as required in design, or cancel

3.8.3 Component Diagrams

Component diagram is also a static view diagram and is used to show the software components, executables, libraries, tables, files and documents of the system to be developed. This diagram has a higher level abstraction relative to class diagram, and components can include one or more classes in itself.

Software, executables, dll files can be components of the system. Components may expose interfaces. They are visible entry points of a component that it makes available to other components for interaction. Interfaces can be any other artifact or class. A component can provide an interface, or require an interface; so two kinds of interfaces exist. There can be components and sub-components of those components. The inner relations between sub-components are shown with “assembly” relation indicating interfaces. A port can be defined for a component, to provide interaction between inner parts of component and outer world. The ports are connected to inner interfaces with “delegate” relation. A dependency relation is used to connect interfaces of components, shown from requiring to providing interface.

SS Entity Ontology diagrams are used to explain physical structure of the system, and will especially be used to introduce simulation system facilities, hardware, software and network and modules that compose these entities. The entities with kind “software” are components of system. The software entity which shows the system to be developed is one component in the system. The parts of that component (connected with p/w relation) are sub-components. Other software entities are other components in the system; they exist especially if developed system is part of a bigger system. SS Work Flow diagrams are the second SS diagrams used to compose component diagrams. Considering each task and the software entity related with that component (as it is stated on notation), interfaces between components are specified. ssInput/Output artifacts are interfaces, the component providing the output has provided interface, and the component using the output has required interface. Actors can be specified as a component providing interface, and actor inputs as mentioned on realize relations are also interfaces for components. If the interface provides relation between a sub-component and another main component, a port is defined.

Work flow diagram of SS CM mostly provides information about software entities which are directly used by actors. Determining the components and relations between them until now, it can be realized that relations between some of the components are not defined. SS ER diagrams may contain information on high level component relations. If available, related information from SS ER diagrams can be used to complete component diagram, just like as done in previous steps.

The following table depicts the elements in CM diagrams and corresponding design elements, to summarize how to use CM to create component diagrams.

Table 6: CM elements used to develop Component Diagram

CM Diagram	Element/Relation	Component Diagram Element
SS EO	Entity (kind=software)	Component
SS EO	Parts of main software entity to be developed	Sub-components
SS EO	Other software entities (if developed system is part of a bigger system)	Other components
SS WF	input/output between tasks of specified components	required and provided interfaces for specified components – define assembly relation between subcomponent interfaces, and dependency relations between component interfaces
SS WF	Actor	component providing interface
SS WF	cmRlz user inputs	required interface of related component
SS WF	subcomponent having interface with other main component	- define port and use delegate relation between subcomponent interface and port
SS ER	relations between other components	required and provided interfaces for components and dependency relations

3.8.4 Deployment Diagrams

This diagram shows how a system will be physically deployed in hardware environment, where components will run and communicate with each other. The node represents a physical or virtual hardware element. An artifact is any product of software development process which is used or produced by system, including process models, documents, executables etc. An association relation represents a communication path between nodes, including multiplicities, name and direction if necessary. Nodes can be placed inside each other, to depict physical hierarchy. Components are shown in the same way as component diagrams, and software components placed inside nodes states that software runs, or deployed on the node.

To develop deployment diagrams, again SS EO diagrams are utilized. The entities with “kind” facility are placed as nodes in deployment diagram. “p/w” relations are used to depict hierarchy among nodes by placing them inside each other.

The next step is to place software components inside nodes. As mentioned before, SS ER diagram may especially be used to depict relation between software and hardware modules, like which software module runs on which hardware. On SS ER diagrams, this information is provided by means of generic relation types shown between facility and software entities. This information is used to place components inside nodes. The artifacts are placed on the diagram by using input/output elements on related SS ER and on SS WF diagrams. Each input/output element is placed as an artifact in the node that the entity which gives output to that element resides. To complete the diagram, associations are drawn between nodes using relations in component diagram. For components having interface with each other in component diagram, an association is added in deployment diagram between nodes carrying those components.

The following table depicts the elements in CM diagrams and corresponding design elements, to summarize how to use CM to create deployment diagrams.

Table 7: CM elements used to develop Deployment Diagram

CM Diagram	Element/Relation	Deployment Diagram Element
SS EO	Entity (kind=facility)	Node
SS EO	p/w relation among entities	Node hierarchy – place nodes inside each other accordingly
SS ER	Relations between facility and software entities	Components inside nodes
SS ER & WF	Input/output entities	Artifacts – place in the node with the entity that outputs it
SS ER & WF	Interfaces between software entities	Associations between nodes – add between each node carrying related entities

3.8.5 Use Case Diagrams

Use case model is used to capture the requirements of the system. Use case diagrams identify the functionality provided by the system, the users who interact with the system (actors) and the association between them. Actor is a user of the system; it is mostly human but can be another application that has interface with and uses the system, but which is external to the system. Use case is a functionality provided by system. Each use case specifies a behavior that the system can perform in collaboration with actors. Use cases define behavior of the system without reference to its internal structure.

Association relation between the actor and use case indicates that the actor participates in use case in some form. Use cases can be connected to each other with include and extend relations. If a use case is related to another use case with "include" relation, it means that first use case contains the functionality of second use case as part of its normal processing. That is, when first use case is called, included use case also runs. The extend relation implies that behavior of a use case may be extended by the behavior of another.

In CM, mission space and work flow diagrams explain the behaviors of the system in relation to actors. But the actors in MS diagrams are internal to the system; they exist

because of related domain operations, not because of the system to be developed. In contrary, the actors in SS diagrams are external to the system. They are actors which use the system and interact externally with the system. Also, the activities they conduct are about external structure of the system, with which they can interact.

Considering these issues, it can be concluded that it is appropriate to use SS diagrams to derive use case diagrams of UML. Actors are obtained from SS organization structure diagram. The roles in SS OS diagram represent external actors of the system. Use cases can be extracted from SS WF diagrams. Each actor specified in previous step shall be handled in WF diagrams. Each task conducted by that actor shall be specified as a use case, and the tasks connected with realize relation to actor shall be connected with an association. Here, the hierarchy of WF diagrams is utilized to specify "include" relations between use cases. A task in the first level is specified as a use case, then the tasks in the second level WF diagram that details the first task are linked to first task with include relation. In this way, use cases are detailed.

The developers shall examine the resultant use case for redundant elements. In WF diagrams, there may be excess tasks used to make work flows clearer which will not be meaningful for use case diagrams, like starting and closing user interfaces. Also, the developer may need to add non-human actors to the system, and can utilize high level facility entities (which are in a higher level than the developed system) to define them. The developer shall specify related use cases for such actors.

In the upper paragraphs, it is concluded that SS WF diagrams can be used to develop generic use cases that are explained in UML. In this situation, MS Mission Space diagrams are left out because they include internal actors and tasks in the system. Blake states that, "Strictly speaking, a Use Case represents one completed set of actions between a user and a software system. However, the term has evolved into a more general usage [66]". As he emphasizes, additional to standard usage, use cases are used for many aims in software development area. Considering this, it is appropriate to use MS MisSp diagrams to develop use cases that describe use cases of internal actors of the system; in this way, MS diagram knowledge is transformed

into and utilized in design. To compose use case diagrams from MS MisSp diagrams, simply all actors in MisSp diagrams are placed in use case diagram, and the missions that are realized by those actors are connected to actors with association line. These internal actors may be individual forces, or teams. Missions connected to each other with include and extend relations are again placed on diagram with same relations, considering lower level diagrams. In this way, in addition to standard use cases, the knowledge of what high level use cases the internal actors conduct is provided.

The following table depicts the elements in CM diagrams and corresponding design elements, to summarize how to use CM to create use case diagrams.

Table 8: CM elements used to develop Use Case Diagram

CM Diagram	Element/Relation	Use Case Diagram Element
SS OS	Roles	Actors
SS WF	Tasks	Use cases
SS WF	Realize relation	Actor-use case association
SS WF	Hierarchy among tasks	include relations
SS EO	High level facility entities	Nonhuman actors – define related use cases
MS MisSp	Actors/roles	Actors
MS MisSp	Missions	Use cases
MS MisSp	Realize relation	Actor-use case association
MS MisSp	include relation	include relation
MS MisSp	extend relation	extend relation

3.8.6 Activity Diagrams

The purpose of activity diagrams is to display sequence of actions that are part of a larger activity. Activity diagrams are mainly used to detail the use cases, but the usage in the project is not restricted to that. It can be used to model business-level functions, or for system-level functions.

Activity diagrams are used in many areas, not only for design. The reason is that they provide a notation that can explain functions of different domains, and they are easy

to read. Likewise, this diagram type is used in conceptual modeling in a similar way. CM work flow and UML activity diagrams are in fact very similar to each other. They have the same objective of modeling sequence of activities, and they almost have the same notation.

An action is a single step of an activity that depicts a function. Action corresponds to task in CM WF diagrams. Actions have pre and post conditions, like tasks. Control flow shows the flow of control from one action to the next, in the same way in tasks. Initial and final nodes are used to show start and end of a sequence in both approaches, and they can be more than one in one diagram. Merge nodes direct more than one control flow to one, according to conditionals. Contrary to this, decision nodes separate control flow to more than one, determining which to be used by conditionals. Fork and join nodes, corresponding to synchronization point in CM, are used to indicate the start and end of concurrent threads.

Expansion and interruptible activity regions can be defined in UML. These do not exist in CM WF and can be added in later phases to detail the execution of actions. CM WF diagram has additional two elements, roles that show who executes the tasks, and input/outputs produced by tasks. These two elements have been utilized in other parts of design, and must be excluded from activity diagrams.

Considering the original aim of activity diagrams, it is appropriate to use SS WF diagrams; as they are explanations of use cases defined before, that are behaviors of system interacting with actors. But activity diagrams can be utilized also in design in many ways, to introduce any function of the system. In MS WF diagrams, internal functions of the system and interactions with internal actors are provided. These MS WF diagrams shall also be used as activity diagrams in design phase, to exploit CM knowledge in design.

The following table depicts which elements correspond to each other in CM WF and UML activity diagrams, and how to use WF to generate activity diagrams.

Table 9: CM elements used to develop Activity Diagram

CM Diagram	Element/Relation	Activity Diagram Element
SS & MS WF	Task	Action
SS & MS WF	Task pre/post condition	Action pre/post condition
SS & MS WF	Control flow	Control flow
SS & MS WF	Initial state	Initial node
SS & MS WF	Final state	Final node
SS & MS WF	Decision point	Merge and decision node
SS & MS WF	Synchronization point	Fork and join node
SS & MS WF	Actors & Realize relation	- discard from activity diagram
SS & MS WF	input/output and relations	- discard from activity diagram

3.8.7 State Machine Diagrams

A state machine diagram models different states of a single class and how that class transitions from state to state. In this way, it can be observed how the entity responds to various events by changing from one state to another.

A state is a situation during which some invariant condition holds. The entity in some state waits for some external event to occur and change state. Initial state denotes the default state of entity to begin with and final state represents the completion of state diagram. Transition from one state to another is represented with transition relation. A trigger event is defined for transition that causes the transition. A condition can be specified for transition. States can be transitioned to themselves.

In conceptual modeling, entities in domain are specified in MS EO diagrams, which are then used to define classes in design analysis. The states of entities and transitions between them are shown in MS Entity State diagrams. Similar to UML state machine diagrams, MS ES diagram includes state, initial and final states, transitions, trigger events and conditions. MS ES diagrams can be used directly in design analysis. At later stages, the developer can add more detailed aspects to state diagrams, like pseudo-states, history states, and concurrent regions.

Other than MS ES diagrams, SS ES diagrams can be helpful in design to understand states of whole simulation system. It increases the understanding of how the overall system works. A transition table is not provided here, because transition from MS and SS ES diagrams to UML state machine diagrams is straightforward.

3.8.8 Other Diagrams and Design Issues

There are other UML diagrams to be developed for which CM will be helpful because of conceptual information it provides, but they include too detailed design decisions that CM cannot provide. The name of those diagrams and a short explanation will be provided here, but no guidelines on how to develop them like previous diagrams will be provided. The developer is expected to manually utilize CM knowledge to develop those diagrams.

Composite structure diagram is one of the structural diagrams. It shows the internal structure of classes, including parts and ports of it and interaction points to other parts of system. The general information on classes that CM provides is utilized in class diagrams, internal structure of class is left for more detailed design activities.

Sequence diagrams show the interaction between objects in the sequence that they occur. Mainly, this diagram depicts the messages transferred between objects, the behaviors executed by objects over time as a result of messages and changes on those objects. Communication diagram carries the same information with sequence diagram, emphasizing on conducted messages rather than time as the difference. Both of these diagrams include detailed design decisions, which also include the time aspect, making it hard to directly use CM. However, work flow diagrams and entity state diagrams of CM can especially be helpful to start developing these diagrams.

Timing and interaction overview diagrams are new in UML 2. Timing diagram integrates sequence and state diagrams, and displays change of state of elements through its lifetime in a different view. Interaction overview diagram integrates activity and sequence diagrams and displays overview of the flow of control of the interactions. Both of these diagrams are integrations of previous diagram types and

include detailed design decisions; so they can be developed in later stages of design by using previous diagrams and CM.

Lastly, in addition to diagrams, high level assumptions and constraints defined during conceptual modeling activity shall also be utilized in design. They will be helpful to define detail levels required in system and understand different aspects of system during design activities.

3.9 Specification of Extensions and Rationale behind Them

To provide the readers a complete explanation of proposed methodology and to prevent ambiguousness of the reading, in previous sections of this chapter, the proposed methodology is explained as a whole. That is, the parts that are already defined by KAMA, and the parts developed in this study as extensions are not explicitly stated. In this section, extensions will be specified and the aim of those extensions to meet study's scope will be explained.

The process of CM development is started to be explained with the section ***3.3 Step 1 – Collect Authoritative Information***. There are two new steps in the process as "develop simulation space CM diagrams" and "develop high level design". Also, some sub-steps are updated. These changes are conducted to be able to integrate extensions inside conceptual modeling process defined by KAMA.

The second step, as explained in ***3.4 Step 2 – Identify Model Elements***, also exists in KAMA. Most elements listed in this section are the same in KAMA, but some new elements are added and some elements are updated.

For the "entity" element described in ***3.4.2.1***, a new property named "kind" is added. This new attribute aims to specify different kinds of an entity; according to which one can understand that, for example, the entity is a force in military domain, or it is a hardware station in simulation system. By means of "kind" property, different entities are easily classified to be used in different parts of simulation design.

Attribute definition of the entity element is updated. First, an attribute is classified as MS or SS. In the original methodology, only MS entities are specified; but in extended method, both are expected to be defined by developers. This is because it would not be possible to fully define an entity without specifying both types of attributes. To be able to define MS and SS models separately, attributes shall be classified accordingly, and entity shall be used in design by using this information. The same arguments are applicable also for behaviors of the entity. Behaviors of the entity that belong to MS or SS shall be defined, and they shall be classified.

The unit of a defined attribute is suggested to be specified, although it is not a must. Mostly, units can be decided in early phases of development. If this information is reflected in CM, consistency between units is easily provided in design.

An initial value can be assigned to attributes. This is not applicable for all entities, but necessary for some to reflect characteristics of the entity. The list of enumerations for the attribute shall be provided. These initial and possible values of attributes are utilized as valuable information in design.

Attributes are classified as fixed or variable, to specify attributes that are assigned for once from DB, or that are updated continuously at run-time. This information eases the understanding of the entity for the user, and it is used in design to specify the type of variable to be defined (like constant, static, derived etc.), and form the mechanism to assign and update the values of attributes.

The object element is added newly in the methodology. It serves many objectives in conceptual modeling. By defining objects of an entity, the specific types of that entity the user requires are stored. Values of some fixed attributes of specific entities are obtained by developers at different times of development. For example, at the beginning of development, user can specify that they require F-16 as a fixed wing vehicle, and can provide the chord length of F-16. The object notation provides developers a storage environment for this kind of data from the beginning of SDLC. Both objects and entities can be utilized in work flow diagrams of CM to describe operations in the system. If the developer wants to depict behaviors of a generic

entity, the entity notation is used. If he wants to depict a specific type of entity, the object notation is used. All information carried with object notation is used in design phase.

The pre-condition and post-condition properties of mission element are detailed, and patterns to identify these conditions are defined. If, when and while conditions are defined. The aim is to detail the work flow diagrams of CM, so that behaviors of the system can be explained in more detail and a detailed design can be reached that utilizes this information.

To achieve similar objective, the definition of control flow is extended not only to include classic control flow (where a task begins when the other ends) but also to consider the disjoint flow and other possible conditions. To provide more flexible usage of decision point and to develop clearer models, decision point definition is extended to include if, when and while conditions. Similarly, synchronization point definition is also extended, to include different types of executions like begin-begin, end-end, begin priority, end priority. In this way, flow of events in the system is depicted and used in design more accurately.

Although task has the same properties with the mission, the concept of “sub-task” is introduced to depict hierarchy between tasks. Sub-tasks are depicted in the same way in design diagrams.

Algorithm is a new entity defined in the method. The algorithm entity is helpful to specify behaviors in the system that are used by more than one entity. They don’t have behaviors on their own. With these properties, algorithms both help users to understand main behaviors of the system, and are used to define interfaces in design. A specific type of relation (used by) is defined to associate algorithms to entities that use them.

Quantity and role definitions are added to generic relationship. This aims to specify information found in many relations in design from conceptual modeling phase; even for generic relationships defined in CM. The usage of quantity is detailed in p/w

relation, and strong p/w relation is defined. These information increase the understanding of CM and is utilized in design diagrams.

At the end of section ***3.4.2 Define model elements***, a short guideline on how to extract main model elements and relations by processing requirements statement is provided. This guide does not aim to provide a full element list; it just aims to be a starting point for developers to build well-defined elements.

Section ***3.4.3 Determine elements as MS or SS*** is a new activity suggested in this study. As KAMA does not deal with simulation space, there is no differentiation step between MS and SS elements, all elements belong to MS. In this study, MS elements, SS elements, and MS elements with SS properties are specified. This is necessary to develop MS and SS diagrams separately, and utilize them in different ways to develop high level design.

Seven diagram types for mission space CM are defined in KAMA and included in the proposed methodology. Moreover, some additional functionalities and explanations describing how to use diagrams are added in this study.

Other than its standard functionality, EO diagrams are utilized to develop environment CM. Environment is an essential part of simulation systems and its existence in the system shall be handled separately. The specification of the environment and its relations with other entities are defined by means of EO diagrams in the proposed methodology, and in this way, environment is also included in the design of the system.

Section ***3.6*** describes the methodology to develop simulation space CM that is proposed in this study, which is an objective of this study. Five of the diagram types that are also used in mission space CM are utilized for SS modeling, but used in different ways. By means of SS diagrams, simulation system aspects are included in CM of the system. All aspects of simulation system, as suggested by many approaches are included, like physical structure of simulation system, the interaction of user and system, the controls in the system and such. They are defined in a

complete way by means of different diagram types, and one can grasp simulation space issues with all perspectives. By developing SS models, a complete CM is developed, and information necessary to develop high level design of the system is obtained.

Lastly, section ***3.8 Step 7 – Develop High Level Design*** explains how to develop high level design by using CM as input; which is the second objective of this study. It is observed that seven of the UML design diagrams can be composed by utilizing information provided in CM. For example, class diagram is developed by integrating information from different diagrams of CM, like MS EO, MS ER, SS EO diagrams. The reader can observe that extensions defined in this methodology, as specified above, are all useful in development of high level design. For example, "kind" property of an entity is utilized to decide using the entity in class, component or deployment diagram.

The developed high level design will be very helpful for the developer, as he will have passed initial hard steps of design activity that requires many basic decisions, and will have a good starting point for the rest of detailed design. By transforming CM data to high level design, information in CM diagrams is carried to design step. This depicts that extraneous information is not collected in conceptual modeling, and conceptual information is utilized completely in the system. This is also a sign to show that conceptual modeling is an effective activity in development life cycle, and CM has such a property as a model that it can be both utilized by user to understand the system, and by developer to understand what user wants, and easily develop high level design by using CM as input. This assures that CM is a bridge between the users and developers of the system.

CHAPTER 4

CM CASE STUDY – SYNTHETIC ENVIRONMENT SYSTEM

In this section, case study results are reported for proposed CM development methodology that is described in ***CHAPTER 3***. The case study research is applied on a synthetic environment project named "Synthetic Environment System (SES)". SES is a simulation system that aims to create a synthetic environment including computer generated forces and virtual environment conditions for Flight Mission Simulators (FMS) to be trained for various operational situations. The project and the case study research are explained in sections below.

4.1 Research Strategy

This part of the study is conducted by using one of the qualitative research strategies, which is the case study. The proposed methodology is decided to be applied on an existing Synthetic Environment System (SES) project.

4.1.1 Case Study Research

Qualitative research technique is applied in many disciplines. It involves "the use of qualitative data, such as interviews, documents, and participant observation data, to understand and explain social phenomena [62]". One of the qualitative research methods is case study research, "which investigates a contemporary phenomenon

within its real-life context [63]". Case study research enables to study the phenomenon "in its natural setting, learn about the state of the art, and generate theories from practice [64]". It also allows "to understand nature and complexity of processes, by answering "how" and "why" questions [64]".

Considering these properties of case study research, this research strategy is selected for the study, to evaluate and validate the suggested methodology. Although most common in social sciences, case study research is also "the most common qualitative method used in IS [62]". Case study research is selected considering the following facts of the study being conducted.

In case study research, phenomenon is examined in a natural setting [64]. Case study "copes with technically distinctive situation in which there will be many more variables of interest than data points [63]" and it relies on multiple sources of evidence. Additionally in case study, "one or few entities are examined; no experimental controls are involved" and "complexity of unit is studied intensively [64]". Likewise, focus of this study is a contemporary issue discussed within its natural domain without any manipulation. A lot of variables are examined and new ones are explored through intensive research. In this way, the issue is tried to be explored in detail, and the theory is enhanced.

4.1.2 Theory Development

"Theory development prior to the collection of any case study data is an essential step in doing case studies [63]". This study examines a contemporary issue in M&S systems, conceptual modeling, and usage of it in a different domain. To offer a solution to this issue, a theory is developed intensively which is based on KAMA. For a detailed explanation of the theory developed, the readers can refer to ***CHAPTER 3 PROPOSED CM DEVELOPMENT METHODOLOGY***.

The case study aims to test the suggested theory. External validity of case studies, that "whether a study's findings are generalizable beyond the immediate case study [63]" shall be provided. The usage of theory in case study is stated as a way of

solving the problem of external validity [63]. The strong theory development and testing in this study also solves the external validity issue.

4.1.3 Single Case Study and the Case Selection

In this study, a single case is used for the case study, to confirm the methodology of CM development as formulated in previous chapter that is suggested for simulation systems, on a case that covers many aspects of the area. The case is critical, including many aspects in the related domain and with many mission space and simulation space specifications. By means of this case study, the proposed methodology will be confirmed and evaluated. The study is also an exploratory study, as the study does not aim to reveal a causal relation in the domain that an event leads to another event; but rather tries to confirm existing hypothesis and propositions and develop them further.

The "Synthetic Environment System" project is chosen as the case for this study. The rationale behind this case selection is the large extent of the project. Synthetic environment simulation systems include many aspects that can exist in different simulation systems; including virtual forces (computer generated forces), advanced mission simulators (operated by man), semi-automated forces (operated by both man and computer), command and control, communications, intelligence, surveillance and reconnaissance devices, weapon systems and other systems existing in battlefield which interact with each other and environment to compose an environment to conduct missions. In author's opinion, if CM can be developed for such a system by using proposed methodology, it can be concluded that methodology is applicable to many simulation systems.

The methodology is aimed to be tested by developing mission space CM, simulation space CM and high level design by using CM as input for SES project. The completed CM may exclude some details of SES project, because of physical constraints of the study and confidentiality reasons. However, this situation will be documented at related sections.

4.2 Design and Pre-Implementation of the Study

In this section, the design of the case study research, basic issues in case study research and the activities conducted before starting methodology testing and development will be discussed. First, research questions are described. Then, general information on SES project on which case study will be implemented is provided.

4.2.1 Research Questions

Defining research questions is seen as the most important step in a research study, as it is the basis for the rest of the studies in the initiating steps. Research questions for this study are also developed considering the scope of the study, and matured through the steps. The research questions identified for this study are as follows.

1) How can KAMA methodology be utilized and what extensions can be added to KAMA methodology to use in a simulation system other than C4ISR?
2) How can KAMA methodology be utilized and extended to develop simulation space CM?
3) How can KAMA methodology and notation be utilized to develop high level design?
4) How does CM development activity using extended KAMA methodology affect requirements analysis, design and development activities in SDLC?

To answer the first question, the proposed extended KAMA methodology to develop mission space conceptual model, that is described in ***CHAPTER 3 PROPOSED CM DEVELOPMENT METHODOLOGY*** will be applied on the synthetic environment project (SES). The outcome of this study, CM of SES project, will be documented in this chapter, resulting CM diagrams will be explained; and how well the methodology suits the project will be evaluated in this way.

To answer the second question, suggested methodology to develop simulation space CM, as explained in ***3.6 Step 4 – Develop Simulation Space CM Diagrams*** will be applied on the same project, SES. The outcome of this study, simulation space CM of

SES project, will be documented in this chapter, resulting simulation space CM diagrams will be explained, and the methodology will be evaluated.

The third question is about design phase of SDLC. To answer the question, high level design for SES project will be developed by using mission space and simulation space CM that are developed before, by applying suggested methodology described in ***3.8 Step 7 – Develop High Level Design***. The design artifact will be evaluated and will be documented in this chapter. The appropriateness of usage of CM in design will be evaluated by considering design artifacts.

To answer the last question, an overall analysis of requirements analysis, CM development and project design activities will be conducted. The extent that CM increases understandability of system and quality of requirements will be discussed. How the development of CM affects total duration of SDLC and possible number of errors in requirements and design will be evaluated. These evaluations and discussions will be conducted according to author's own experiences obtained during development of SES project and development of CM for SES. Also, the ideas of the rest of the SES development team on the project and conceptual modeling will be utilized.

4.2.2 General Information on SES Project

In this section, general information on SES will be provided, though all details and real name of the project will not be given as the information is confidential. More detailed information required for CM development will be provided in following sections.

In simulation domain, synthetic means combining constructive, virtual or live entities to provide a synthetic battlefield. Mostly, user is immersed in the application; and realistic or physics based simulation of critical objects and interactions are required. Synthetic environment applications have increasing importance in M&S field. Chapman stresses the importance on his study where he develops a CM for a DMT (Distributed Mission Training) Center which links together advanced simulators (F-15C), CGF, selected real world command and control systems, and fighter, bomber,

ISR sites [2]. Ayres and Atherton have also studies on developing models for synthetic environments [65], [50].

SES project is a simulation system that aims to create the synthetic environment that is needed for FMS to conduct various military operations under different conditions. FMS's are full flight simulators for two kinds of rotary wing airborne vehicles. Synthetic environment includes different kinds of players which are Computer Generated Forces (CGF). CGF means that, these players can behave near real and use their player systems (like sensors and weapons) in the battlefield to conduct rules assigned to them in the scenario. No human interference is required for them to behave meaningfully during scenario execution.

Types of players are ground vehicles, fixed platforms, rotary and fixed wing airborne vehicles, surface and underwater vehicles. These players have advanced dynamics that take into consideration various effects in environment (like meteorological conditions, terrain conditions), collisions and damage levels. Advanced rules can dynamically be created and assigned on players that cover many possible conditions and behaviors.

These players may use many player systems simulated in the system; mainly sensors (radar, electro-optics etc.), countermeasures (chaff, flare, jammers etc.), communication systems (radio, direct link etc.) and weapons (missile, rocket, bomb, bullet etc.). Also, environmental conditions are taken into consideration. The operational field is Turkey, which is simulated with high resolution GIS files, different terrain types and cultural features on it. Weather conditions like temperature, pressure, wind, precipitation etc. are simulated in the system, and these conditions are taken into consideration by players and player systems. Also underwater conditions and sea waves are considered by sea platforms.

Looking at simulation system properties, SES works at SE Control Center (SECC) which has two kinds of stations, SE Manager Station and SE Simulation Stations. SE Manager Station has interfaces to define player systems, players and scenarios, to start a scenario and to watch and control an ongoing scenario. SE Simulation Stations

have interfaces to edit scenarios and to watch ongoing scenario. SE Simulation Engine at run-time works on SE Simulation Stations.

SES project development team is composed of five to nine software engineers, changing through life cycle. One of them works as the technical project manager. Other than two quality engineers in the company supporting project development activities, two or three test engineers works before and during delivery phases to the customer. The total development time of the project is three years; two years for standalone development of SES, and one year for integration of SES with the main system.

4.2.3 Data Collection Activities, Reliability and Validity

Reliability assures a later investigator to "arrive at the same findings and conclusions [63]" when the same case study is conducted. To provide reliability in this study, sources of information and the guideline on how to conduct the case study are determined by the author. Format for the report is specified as classic narrative type to describe and analyze the case.

Sources in the literature, as listed in ***CHAPTER 2***, are reviewed to understand conceptual modeling field and current situation in the field. About two months of effort is spent by the author of this study to understand KAMA and make basic decisions on how to use and enhance it to cover deficiencies of conceptual modeling. Five face-to-face meetings, which are semi-structured interviews, are arranged to discuss KAMA with MODSIM personnel; other than short daily discussions. Discussions with these personnel are carried on to discuss the methodology and the case study application on SES project.

As a result of these activities, the sources of information are collected and utilized in this study as described in following paragraph.

Two groups of documentation are collected and utilized in this case study research. First group is the documentation found as a result of literature review. Second group is the documentation about the project for which case study is conducted, SES. This

documentation includes development documents of the project, like user specifications, system requirements, system design documents, manuals. Other documentation about this project involves materials on synthetic environment domain knowledge.

Discussions are also conducted with SES development group, to discuss the possible implementation and benefits of conceptual modeling in the project. The author of this study has participated in software development process of SES project. The author has used her direct and participant observations about the project during conducting of this case study.

To increase reliability of the case study, the author of this study has created a folder structure for all soft copies in computer environment, and used a versioning system for reports created by her. For hard copies, the author created an index system to easily reach the searched document, and used color coding to determine the importance and define notes.

Construct validity needs to be provided in case study research during data collection activities. Usage of multiple sources of evidence for the same phenomenon provides construct validity for case studies. In this study, by using different types of sources, including documentation, observations and interviews, same phenomenon is observed from different perspectives, and all are used for the same objective. Also, the chain of evidence is maintained, by preparing a clear report, providing references to sources in the report, and providing explanations on the study and questions.

4.3 Case Study Implementation and Report

In following sections, the process of CM development will be conducted on the case project, SES, as suggested in ***CHAPTER 3 PROPOSED CM DEVELOPMENT METHODOLOGY***; the process and the resultant products will be explained step by step. Other than the obtained domain knowledge, the main input to develop CM is system requirements document of the project. The reason for that is, the most mature

requirements document for the project is this document, from which a complete CM can be developed. The documentation of developed CM and discussions will aim to answer research questions one by one, as specified in ***4.2.1 Research Questions***.

4.4 Case Study Implementation - CM Development for SES

In the following sections, MS and SS conceptual models and high level design for SES project will be developed, following the steps as described in ***3.1 CM Development Process***. CM is developed by using a CASE tool. It is not important which CASE tool to use, as long as the developer utilizes UML notation as CM language notation. In this study, a CASE tool which is widely used in industry and that the author believes to have a good user interface is used. A profile is created on the tool to define CM elements from UML elements. For example, entity element is defined from a class element. In this manner, all elements and relations of CM are defined and used in diagram development activities.

4.4.1 Step 1 – Collect Authoritative Information

Authoritative information is required to understand SES and define details on the area to model and implement the system. Main objective of SES is to create synthetic environment that enables execution of military operations for air vehicles under different situations. Considering general information on SES provided in section ***4.2.2***, the boundaries of the objective are ground, air and sea players, their dynamics and collision and damage physics, different types of player systems and different environment conditions. The simulation context shall be examined considering these boundaries. Some of the topics on which authoritative information shall be collected in simulation context are as follows;

- Military operation types and tactics for rotary wing airborne vehicles
- Dynamics for airborne vehicles
- Dynamics for ground vehicles
- Dynamics for sea platforms
- Physics for collision and damage

- Signal propagation in air under different conditions (for sensors, jammers etc.)
- Weapon ballistics
- Artificial Intelligence and doctrine creation for military operations
- Acoustic modeling (for sonar)

During development of the project, authoritative information is collected for such topics in simulation context, examining many military sources and contacting customer. This information is to be used for conceptual modeling (in our case), and is also used to mature requirements. Sources of information shall also be kept.

4.4.2 Step 2 – Identify Model Elements

In this section, model elements for SES will be identified by examining requirements statement of SES, according to guidelines provided in section ***3.4 Step 2 – Identify Model Elements***. There are about four hundred requirements for SES in total. More than thirty of the requirements will be listed below; and conceptual model elements in those set of requirements will be identified with their properties. In this way, it is aimed to provide more detailed information on SES, and to provide examples on how to identify elements. For the rest of the elements used in CM, related requirements will not be written in this study, but elements identified by the author will directly be used in CM diagrams in steps 3 and 4.

In this section, requirements for SES will be given as sets, then model elements extracted from those requirements will be explained under that part. Following is a requirements set for SES that describes main properties of system.

1. **REQUIREMENT SET**

REQ 1 Synthetic Environment System (SES) shall be a part of Flight Simulator Center System (FSC), together with Flight Control Center (FCC).

REQ 2 The Flight Control Center (FCC) shall be composed of hardware and software components.

REQ 3 FCC hardware shall be composed of a FCC station and Apache and Cobra Mission Simulator hardware.

REQ 4 Flight Control Software (FCS) shall run on FCC, shall be composed of Flight Management Software (FMS) and Mission Simulation Software modules.

REQ 5 Flight Management Software shall run on FCC station and Mission Simulation Software shall run on Mission Simulators.

Until now, the main system under which SES resides, and other parts of that system are explained. Let's list the elements as in the figure below, by means of underlined phrases.

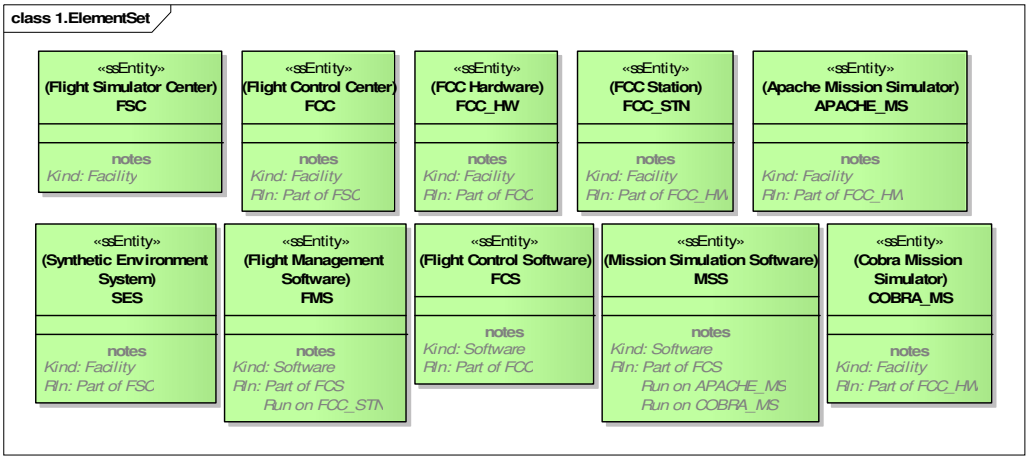


Figure 3: Model Elements for 1. Requirement Set

As all elements are about the simulation system, they are all SS entity elements, and depicted with "ssEntity", and they are shown in green color. The hardware components of simulation system are tagged with kind "facility", and software components are tagged with kind "software". A short and unique code for every element is provided. For the moment, before starting diagramming, relations of elements are depicted as notes.

2. REQUIREMENT SET

REQ 6 The Synthetic Environment System (SES) shall be composed of hardware and software components.

REQ 7 SES software shall create Synthetic Environment for the Mission Simulators to be able to execute operations on military perspective.

REQ 8 SES hardware (named SE Control Center) shall be composed of SE Manager Station and SE Simulation Stations.

REQ 9 SES Software shall be composed of SES Offline and SES Run-Time software modules.

REQ 10 SES Offline module shall be composed of SES DB Edit, SES Scenario Edit and SES Replay modules.

REQ 11 SES Scenario Edit module shall be composed of Main Scenario Edit and Tactical Map modules.

REQ 12 SES Run-Time module shall be composed of SE Runtime Management, Runtime Simulation Engine and Runtime Tactical Map modules.

REQ 13 SES DB Edit module shall run on SE Manager Station.

REQ 14 SES Scenario Edit module shall run on SE Manager Station and SE Simulation Stations.

REQ 15 SES Replay module shall run on SE Manager Station.

REQ 16 SES Run-time Management module shall run on SE Manager Station.

REQ 17 SE Run-time Tactical Map module shall run on SE Manager Station and SE Simulation Stations.

REQ 18 SE Simulation Engine module shall run on SE Simulation Stations.

The following figure illustrates the elements extracted from these requirements. All elements are again SS entities. Hardware stations and software components of SES are described. Notice also that ***REQ 7*** does not mention any elements, as it explains the objective of the system.

3. REQUIREMENT SET

REQ 19 Operation Senior Major shall manage FSC system to arrange operations between FCC and SES.

REQ 20 FCC Major shall operate FCC to manage Mission Simulators.

REQ 21 Mission Simulator Captains shall operate Mission Simulators and get trained.

REQ 22 SE Manager Major shall manage SES to conduct SES Offline and SES Run-time activities, and to command SE Simulation Station Captains.

Figure 4: Model Elements for 2. Requirement Set

REQ 23 SES Simulation Station Captains create and edit scenarios on SES Simulation Stations.

REQ 24 SES Simulation Station Captains monitor ongoing scenario on SES Runtime Tactical Map.

This requirement set gives information on actors and roles in the system. The actors belong to MS, and to differentiate from SS elements, they are shown in pink color. Roles belong to SS, as they conduct operations on SS facilities. The activities the roles conduct are shown as notes on elements for the moment. During diagramming, this information will be utilized as tasks in WF diagrams.

4. REQUIREMENT SET

REQ 25 SE Manager creates and edits the following platform types on SES DB Edit Module.

Airborne platforms – Fixed Wing and Rotary Wing

Ground platforms – Tracked and Non-Tracked

Fixed platforms

Surface platforms – Ship and Fixed Surface

Subsurface platforms – Submarine and Fixed Subsurface

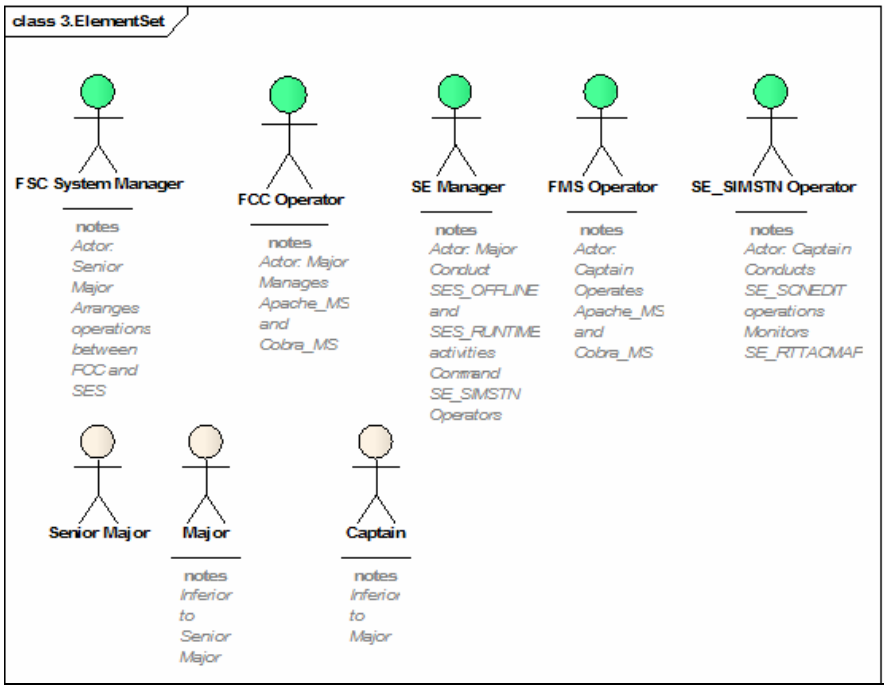


Figure 5: Model Elements for 3. Requirement Set

REQ 26 SE Manager creates and edits the following player systems on SES DB Edit Module.

Sensors – Radar, Electro-optics, Warning Receivers, Laser, Sonar

Countermeasures – Chaff, IR Flare, Jammers, Acoustic Decoy

Communication Systems – Radio, Direct Link

Weapons – Missile, Rocket, Bomb, Gun, Torpedo, Depth Charge

REQ 27 Player systems are assigned to platforms as part of platforms on SES Scenario Edit Module.

REQ 28 SE Manager creates and edits the following types of rules for players.

Opponent selection, and action rules of following types: maneuver, formation, use weapon, use countermeasures, use sensors, use communication systems.

REQ 29 Rules are assigned to platforms on SES DB Edit Module.

REQ 30 SE Manager and FCC operator creates and edits scenarios on SES Scenario Edit Module, by using Player System, Platform and Rule input files created by SES DB Edit Module, and by specifying the terrain.

REQ 31 CGF's of every type of platform can be created in scenario by adding the following properties in SES Scenario Edit Module.

> Scenario player name, player status (active, deactive, killed), force type (blue, red, neutral), player latitude, player longitude, player altitude/depth, player heading, player speed, damage level, activation delay time, fuel level, mission route waypoint latitude, mission route waypoint longitude, mission route waypoint altitude/depth, mission route waypoint speed, formation status, radio frequency

REQ 32 SE Manager selects one scenario and starts the scenario at SE Runtime Module by using SE Run-time Management module.

REQ 33 Flight mission simulators join to scenario as a player with the following properties.

> Scenario player name, player status (active, deactive, killed), force type (blue, red, neutral), player latitude, player longitude, player altitude/depth, player heading, player speed

This part of the requirements provides information about military domain elements that exist in the system. We understand that different platform types are defined in the system, and CGF types in the battlefield are created for different kinds of platforms by defining scenario parameters for them. There is an inheritance relationship for platform types and CGF types. The elements identified about the platform types are shown in ***Figure 6*** below. They are all MS entities and shown in pink color. Attributes are determined for CGF entity, some of them belong to SS, and some belong to MS. They are grouped on element notation. Notice that although platform types are tagged with kind "information", CGF is tagged with kind "Force". The reason is that, platforms exist as definitions in DB and they do not have existence in battlefield. However, CGF entities are placed in battlefield (scenario) and have existence and capabilities on its own.

Some of the requirements inform us about the player systems that exist in the system. They are all MS entities, as they are part of military domain. Player system is a part of platform, and all player system types inherit from generic player system entity. The elements about player systems are shown in ***Figure 7*** below. Similar to

platforms, the kind is “information” for generic player systems as they are only definitions. It is “equipment” or “material” for specific player systems that have capabilities on their own.

Some input/output files that are used by simulation system are specified in the requirements. They are depicted as SS entities in ***Figure 8***. Lastly, the rules specified as part of platforms are described in requirements. The associated elements are depicted in ***Figure 9***. The requirement set above includes other specifications about behavior of entities and interaction between actors and entities in the system. Those specifications will also be utilized as tasks in WF diagrams and relations in various diagrams.

5. REQUIREMENT SET

REQ 34 Platforms shall have the following properties.

Platform name, visual model name, platform length, platform width, platform height

REQ 35 All player collisions shall be calculated according to its bounding radius.

REQ 36 Fuel consumptions of moving players shall be calculated according to player’s speed, directly proportional to a fuel consumption rate.

REQ 37 On collision, all moving players and weapons shall decrease damage level of the entity which it has collided, proportional to damage producing level the entity has and the collision speed.

REQ 38 Maneuvering abilities of the moving player shall be decreased by a maneuvering degradation coefficient, proportional to damage level of the platform.

REQ 39 All airborne platform air dynamics shall be modeled by using the following properties.

Maximum roll rate, default roll rate, maximum climb rate, default climb rate, maximum dive rate, default dive rate, maximum acceleration, default acceleration, maximum deceleration, default deceleration, maximum speed, minimum speed, maximum altitude, player empty mass

REQ 40 Airborne platform dynamics shall be affected from wind.

REQ 41 Airborne players shall complete refuel/rearming at refuel/rearm time.

REQ 42 Airborne platforms shall have radar mean signature to be used by radar.

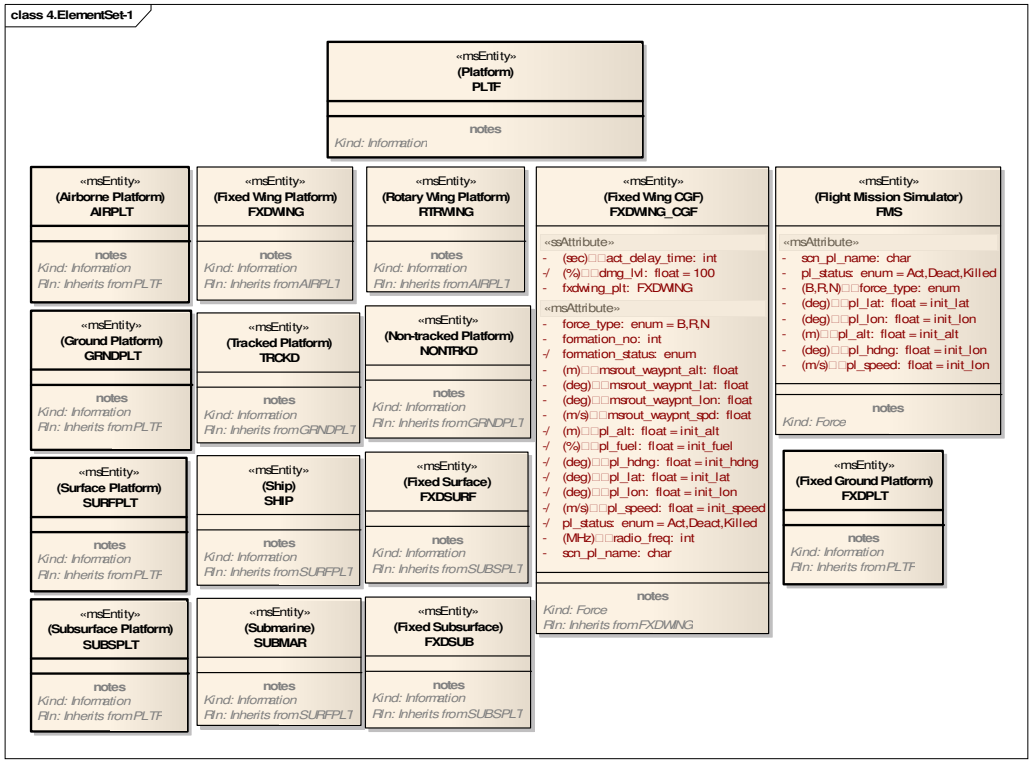


Figure 6: Model Elements about platform types for 4. Requirement Set

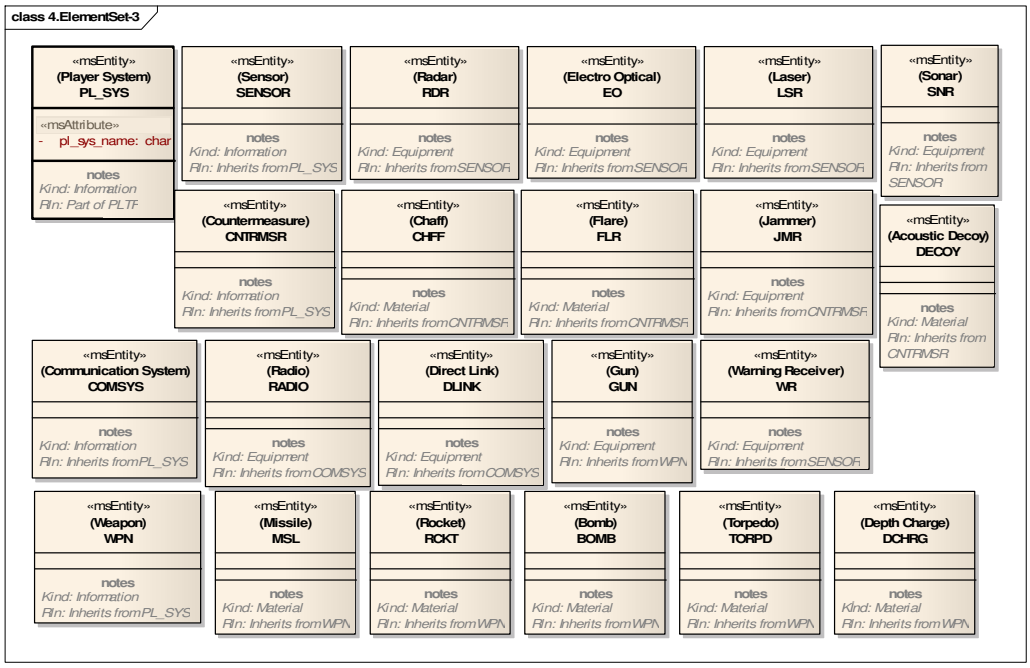


Figure 7: Model Elements about player system types for 4. Requirement Set

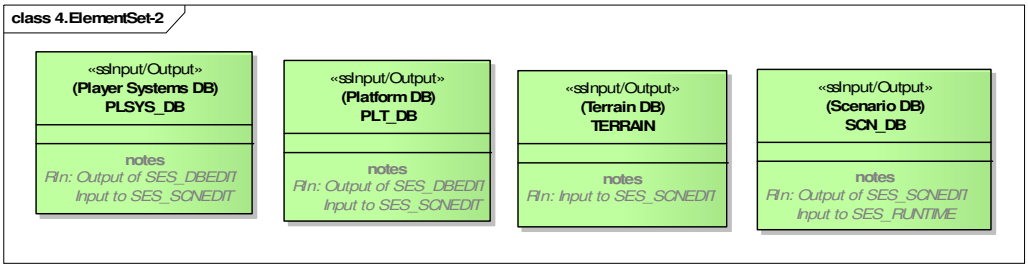


Figure 8: Input/Output Model Elements for 4. Requirement Set

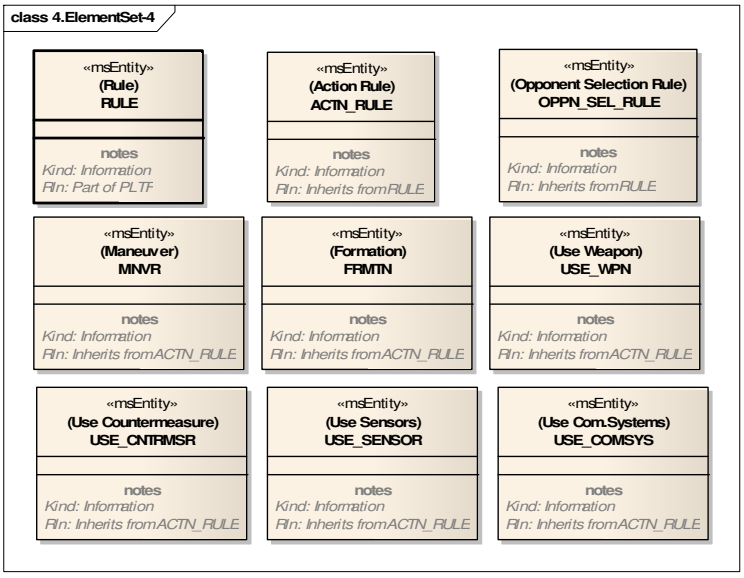


Figure 9: Model Elements about rules for 4. Requirement Set

REQ 43 Airborne platforms shall have thermal mean signature to be used by thermal electro-optics.

REQ 44 Airborne platforms shall have player body temperature to be used by thermal electro-optics.

REQ 45 Airborne platforms shall have contrast ratio to be used by visual electro-optics.

REQ 46 Airborne platforms shall have player reflectivity to be used by laser.

REQ 47 Airborne platforms shall have passive acoustic signature to be used by sonar.

REQ 48 Fixed wing platform air dynamics shall additionally consider the following properties.

Maximum g, default g, maximum pitch rate, chord length, drag coefficient.

REQ 49 Rotary wing platforms air dynamics shall additionally consider the following properties.

Maximum turn rate, default turn rate, blade_length, maximum roll angle, maximum pitch angle, engine power,.

Firstly, this set of requirements informs us about the detailed attributes and behaviors of platform entities that are previously defined. By using the above information, the attributes and behaviors of entities PLTF, AIR, FXDWING, RTRWING, FXDWING_CGF, RTRWING_CGF can be specified in detail, as shown in ***Figure 10***. Notice that as platforms are just definitions that do not exist in battlefield, they don't have behaviors. Behaviors of CGF entities, collision, fuel consumption and such capabilities are depicted. As input to these behaviors, the algorithm used by the entity is written. In fact, the other elements identified by using requirements above are algorithms. The algorithms used by platform entities are depicted in ***Figure 11***. Also, the algorithms that are used by player systems are started to be described in these requirements. Used by relations are written in algorithms, to mention the platforms that utilize those algorithms. Also, the behavior of CGF entity that uses that algorithm is specified by writing the name of the algorithm as input to behavior that uses that algorithm.

For this section, this is all the examples for requirements and extraction of model elements from these requirements. SES project is explained in detail, the objectives of the system, how the system works and main model elements are studied. Many types of model elements are identified. In the next section, CM diagrams will be developed using these elements. The rest of element identification activities will not be described here, but identified elements by the author will be directly used in diagrams.

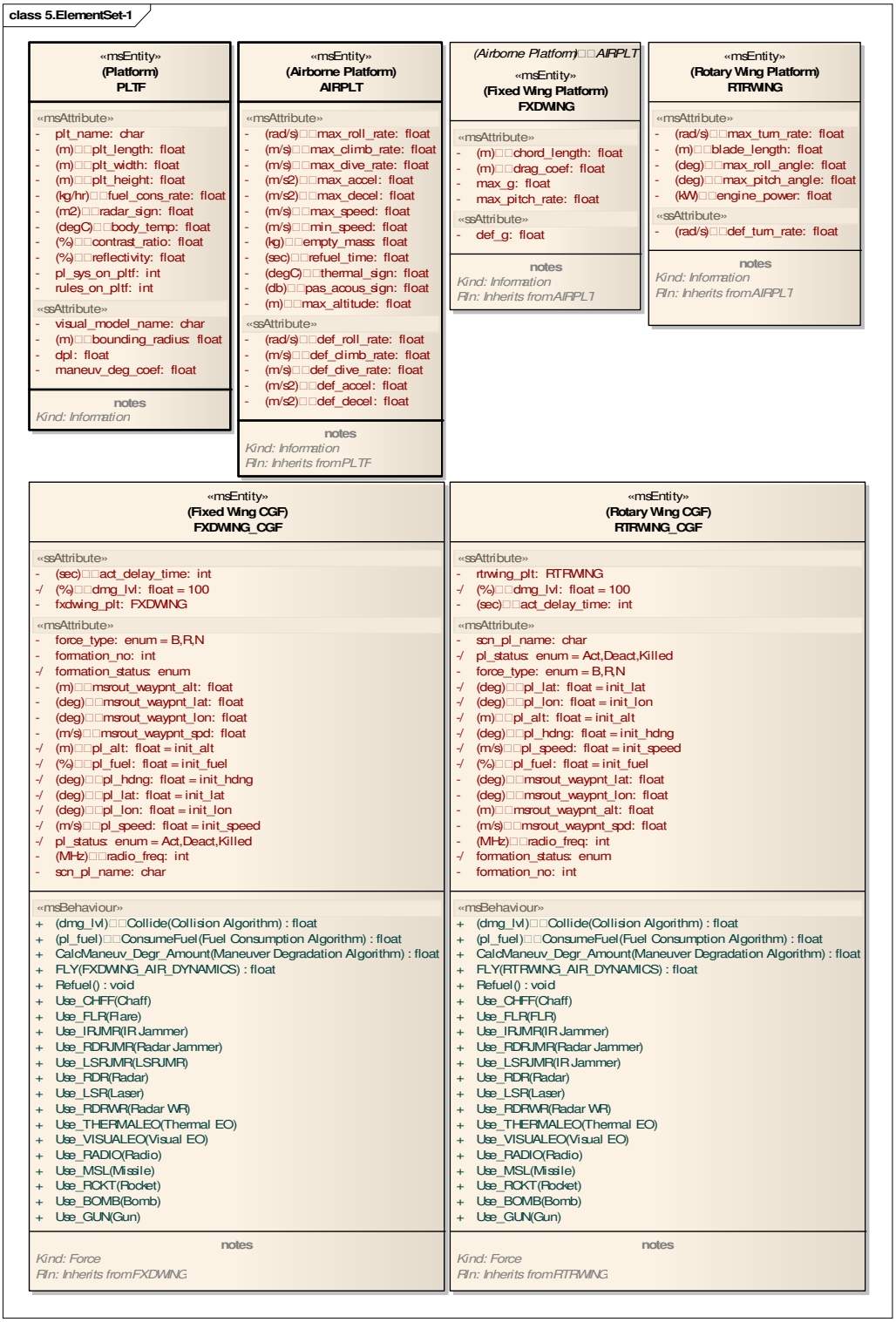


Figure 10: Detailed platform entities for 5. Requirement Set

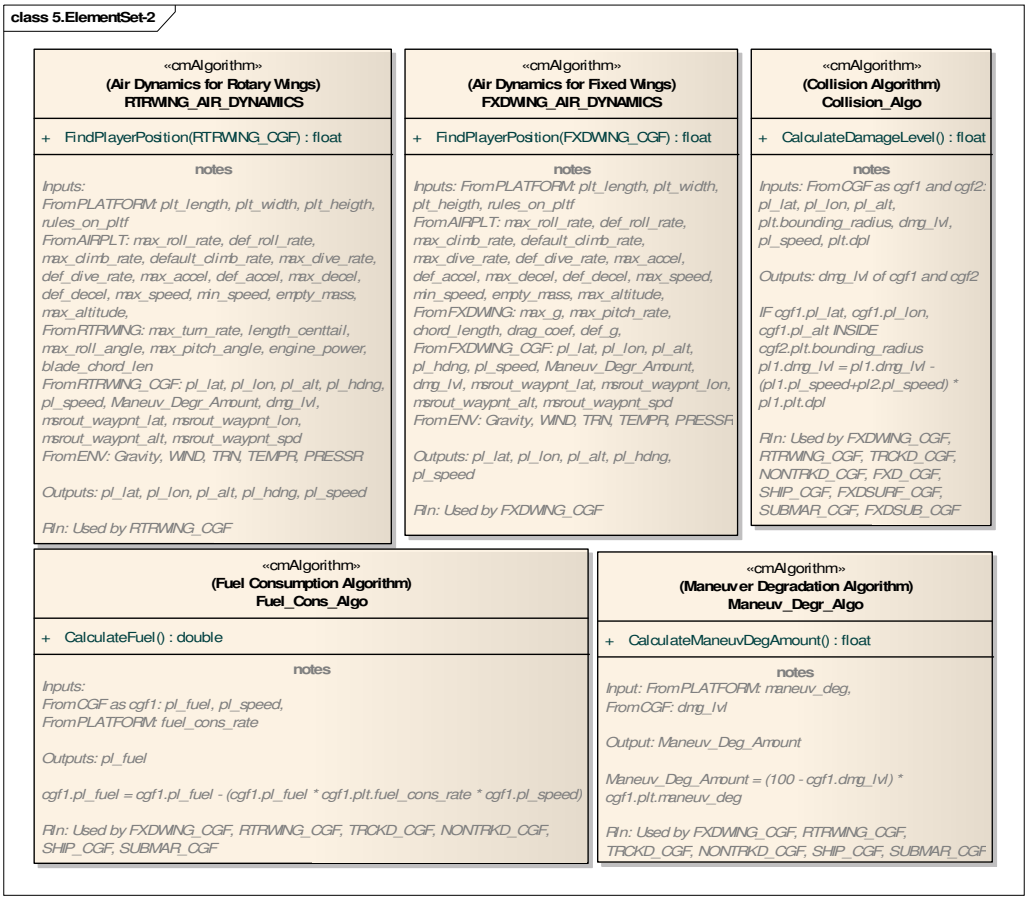


Figure 11: Algorithm entities for platforms in 5. Requirement Set

4.4.3 Step 3 – Develop Mission Space CM Diagrams

In section ***4.4.2 Step 2 – Identify Model Elements***, some of MS model elements are already identified. In this section, using previously defined elements and adding new ones, MS conceptual model diagrams will be developed by using the methodology described in ***3.5 Step 3 – Develop Mission Space CM Diagrams***.

Although some of the elements and relations are defined in previous sections, in the diagrams, there are many elements which are not previously defined or for which requirements are not provided. The reason is physical limits of this document and confidentiality concerns. As the reader of this study grasps an understanding of main features of the system with the descriptions provided (starting with section ***4.2.2***),

and the requirements provided until now, it is assumed that the reader will understand the rest of the system by observing CM diagrams, and there is no need to provide every requirement. All types of diagrams are provided in below sections that compose CM for MS of SES.

All MS diagrams created for SES are not placed on this study, or some properties and attributes on some diagrams are hidden. This way is followed to depict SES CM, because of physical constraints and confidentiality concerns of the study and to decrease complexity. All of the CM diagrams with full properties exist on the technical report published by the author of this study, which includes full conceptual model of SES. The technical report can be found from reference [67], and this report will be referenced at diagram explanations below.

4.4.3.1 Entity Ontology Diagram

The following is the MS Entity Ontology diagrams developed for SES, utilizing the previously identified model elements and adding other necessary elements. Diagrams from ***Figure 12*** to ***Figure 23*** depict mission space EO diagram models for SES. All EO diagrams with full properties can be found at [67].

In the first diagram shown in ***Figure 12: MS EO Diagram 1***, the highest level entities that exist in the system are depicted. For the PLTF, PL_SYS, RULE and ENVRN entities, their kind is specified as information, as they are yet descriptions of entities. All entities are shown in light pink color as they are all MS elements; and multiplicity numbers are determined. Attributes of entities are classified as MS and SS attributes. The details of these elements will be provided in lower level diagrams. Because attributes of PLTF and FMS entities were already depicted in ***Figure 6*** and ***Figure 10***, they are not visualized on this diagram; to decrease complexity of the diagram.

The FMS is shown as a separate entity in the system. This is the entity that is not simulated by SES, but that exists in SES (via HLA network as defined in SS diagrams). In this way, FMS is reflected in SES directly as a force entity. Because

SES does not simulate FMS, the behaviors of FMS that only interests entities in SES are defined in SES conceptual model.

In ***Figure 13: MS EO Diagram 1.1***, all types of PLTF entities that are derived from PLTF are shown. There are five types of platforms in the system. All have many MS attributes that define their identities, dynamics and other behaviors. Also, SS attributes are defined that are used for their existence in SS. These entities are still information, that does not have any behaviors, and all attributes are fixed that are assigned from DB. In this diagram, the attributes of entities are not shown; and full version can be found at [67].

In ***Figure 14: MS EO Diagram 1.1.1***, many details are provided for the first type of platforms, AIRPLT. First, it is observed that two kinds of platforms as FXDWING and RTRWING are derived from AIRPLT. These platforms are composed of many physical parts, which are shown with p/w relations. The next stage is derivation of FXDWING_CGF and RTRWING_CGF from FXDWING and RTRWING entities. The important point is that, FXDWING and RTRWING platforms are of kind "information"; but FXDWING_CGF and RTRWING_CGF gain their identity with the kind "force". Former is just a piece of information on related kind of platform, but the latter has an existence in battlefield. The "CGF" entities are the ones that have behaviors, and that have variable attributes that are calculated and updated at runtime. That is why they are "forces". They all describe what that kind of player can do, including their dynamic behaviors and the player systems that the player can use. At the last stage, the formations that are made up of players are depicted. Attributes and behaviors are not shown on this diagram, but they can be observed from ***Figure 10*** and [67]. It is possible to observe attributes that are given initial values, and attributes for which enumeration values are defined. For some of the behaviors, like "Collide", name of an algorithm is provided as input to behavior. It is understood that this algorithm is utilized by this entity to conduct "Collide" behavior. The relation between the algorithm and entity will be depicted in detail in entity relationship diagrams.

Figure 15: MS EO Diagram 1.1.1.1 is different from other diagrams, as it includes objects rather than entities. In this diagram, possible objects that can be derived from force entities are shown. In this way, what real actors will exist in battlefield can be observed, and the values of their fixed attributes can be stored. The attribute values assigned are not shown in this diagram, but examples for some players can be found at [67]. The rest of the elements do not include attribute values, but just indicate other objects of airborne forces.

Similar EO diagrams exist that detail GRNDPLT, FXDPLT, SURFPLT and SUBPLT entities, like the diagrams in ***Figure 14*** and ***Figure 15*** for AIRPLT. These diagrams are not included in this study, but they can be found in [67]. To summarize, there are two kinds of GRNDPLT from which players are derived, tracked (TRCKD_CGF) and non-tracked (NONTRKD_CGF). These players both have attributes and specific behaviors associated with related algorithms; and objects for these players are stated. FXDPLT entity has one type of player, FXD_CGF. SHIP_CGF and FXDSURF_CGF entities are inherited from SURFPLT entity. Lastly, SUBMAR_CGF and FXDSUB_CGF entities are created from SUBSPLT entity type. All of these entities have their own attributes and behaviors; parts for them are identified; and related objects are stated. It is observed that, although FXDSURF_CGF is a fixed player, it can have player systems and can use them. FXDSUB_CGF has no special behaviors or player systems to use.

Starting from ***Figure 16: MS EO Diagram 1.2***, player systems that exist in the system are started to be explained. Four types of player systems are derived, for which detailed systems will be specified in latter diagrams. Main player system types in the system are sensors, countermeasures, communication systems and weapons.

In ***Figure 17: MS EO Diagram 1.2.1***, player systems which are derived from sensor are provided. The MS and SS attributes for all sensor types are not provided on this diagram, but on [67]. Starting from this level, sensors have the kind “equipment”, but behaviors for them will be defined in lower level diagrams.

In ***Figure 18: MS EO Diagram 1.2.1.1***, the entity RDR's structure is explained in more detail. The attributes of entities can be found at [67]. RDRMOD is shown as a part of RDR. It can be observed that RDRMOD has active and passive types. RDR's functionality is carried out by RDRMOD entities. There can be many RDRMOD entities as a part of RDR entity. But at one moment in run-time, RDR can use one of them. This is understood by means of variable "rdrmode_current" attribute. The behaviors of RDRMOD entities use radar algorithm as input. RDRMOD types can use these behaviors on specified player types. In this way, one can understand which platform types radar can detect. Behaviors of each radar mode can be applied for FXDWING_CGF, RTRWING_CGF, TRCKD_CGF, NONTRKD_CGF, SHIP_CGF, FXDSURF_CGF and SUBMAR_CGF players. The details for sensors electro optics, laser, warning receiver, and sonar are also provided in a similar manner, which can be found at [67].

As the player types of sensors have finished, the next one is countermeasure entities as shown in ***Figure 19: MS EO Diagram 1.2.2***. There are four types of countermeasures. CHFF, FLR and DCY are of material type, as they are ejected and used for once. JMR types are equipment. Communication systems are shown in ***Figure 20: MS EO Diagram 1.2.3***. There are just two types of communication systems, which both have the same behavior of sending messages. As the last player system, weapons are shown in ***Figure 21: MS EO Diagram 1.2.4***. There are six types of weapons in the system that can be used by air, ground and sea players. Some of the weapons have improved properties, like missiles and torpedoes carrying different types of seekers. Attribute details for these diagrams can also be found at [67].

Figure 22: MS EO Diagram 1.3 explains the entity "RULE". The rule is an entity that is defined as part of platform. There may be two kinds of rules, action rule and opponent selection rule. Both kinds of rules have a rule condition attribute that defines conditions which are expected to occur under different circumstances. ACTN_RULE has "response" parameter that defines the action to be done when condition is true. OPPN_SEL_RULE does not have response parameter, as the single response as a result of meeting the condition is setting prime opponent (PO).

Rules are the parts of players that define how the players will behave, in a way adding artificial intelligence to virtual players. It is a specific entity for synthetic environment simulation, but in a way the situation is similar to real players having rules defined by their commanders. That is why the rules are defined as MS entities. The rules are important part of CM. There may be many kinds of action rules according to type of response the player conducts. The player can conduct a maneuver, use any of its player systems with predefined constraints, and change behavior in formation. In this way, virtual forces can behave real-like in battlefield on their own, and adapt to changing conditions at run-time.

The definition of rules in DB is an SS issue, as a result, the description is provided in SS diagrams. Some possible examples of rule elements are also provided on those diagrams. Rules may involve all kinds of possibilities that can occur in battlefield, that are explained on these MS diagrams; including states, an attribute reaching a value, an event etc.

Figure 23: MS EO Diagram 1.4 provides all details about a separate entity in the system, the environment (ENVRN). Handling environment modeling as a separate issue in conceptual modeling is discussed in ***3.5.1 Entity Ontology (EO) Diagram***. In this study, environment modeling in CM is given special importance and environment is modeled as a separate entity, including meteorological, oceanographic and terrain aspects. Modules of environment are modeled as separate entities, with their attributes. In this way, the interaction of other entities in system with environment is easily depicted. As shown in the diagram, mainly ENVRN is composed of three parts, meteorological, oceanographic and terrain entities.

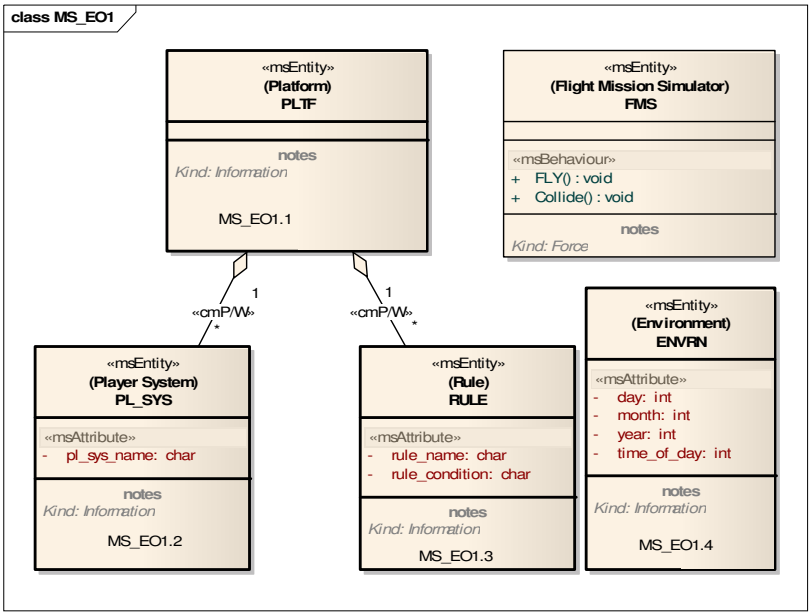


Figure 12: MS EO Diagram 1

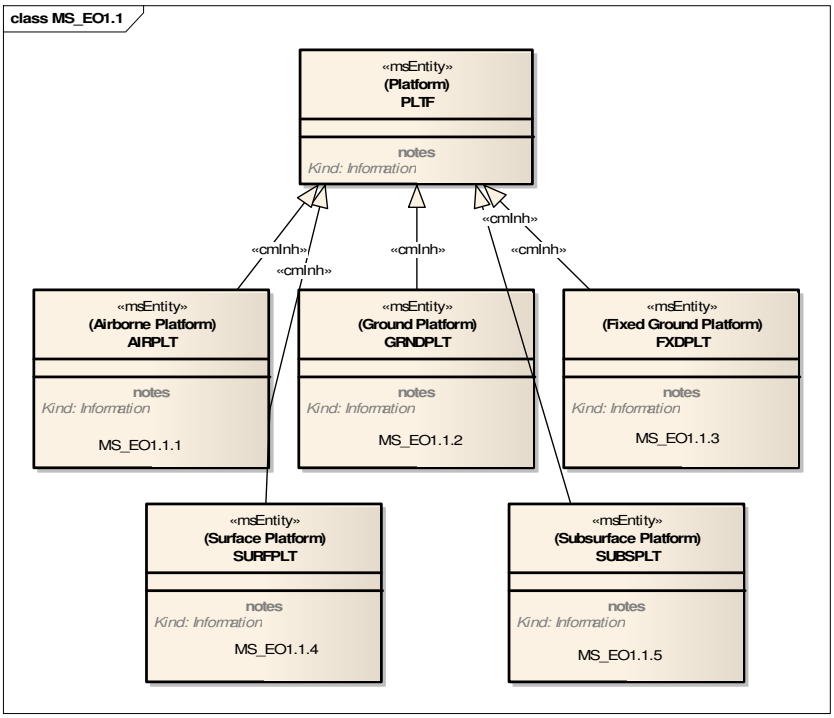


Figure 13: MS EO Diagram 1.1

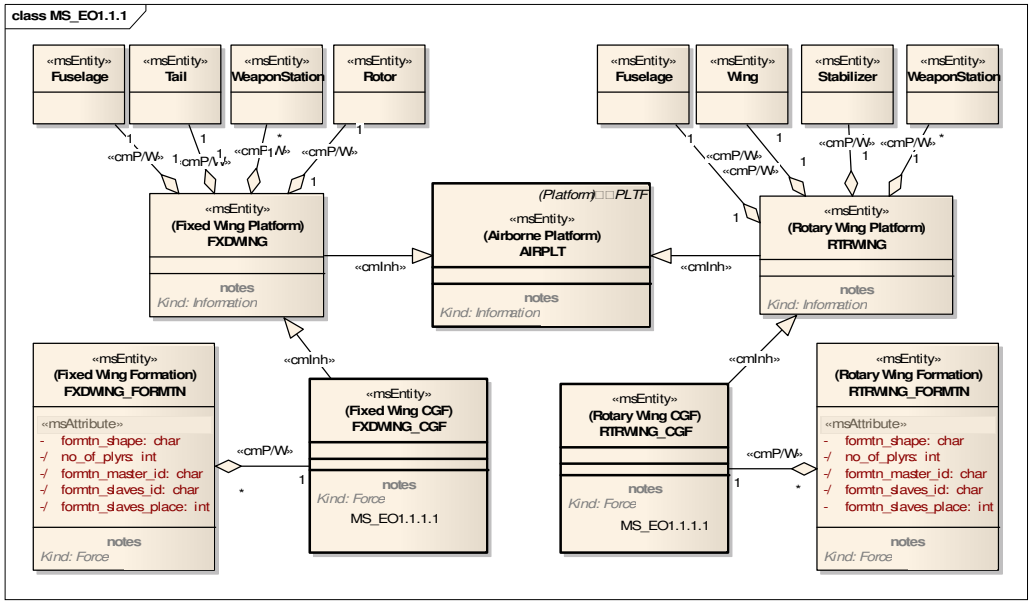


Figure 14: MS EO Diagram 1.1.1

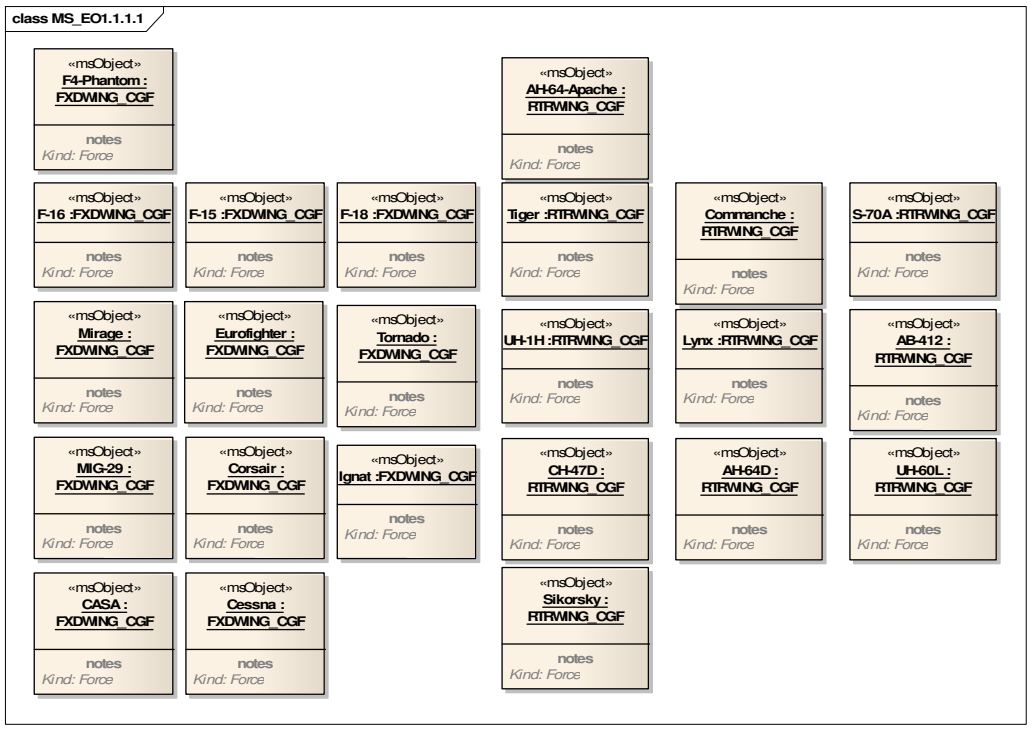


Figure 15: MS EO Diagram 1.1.1.1

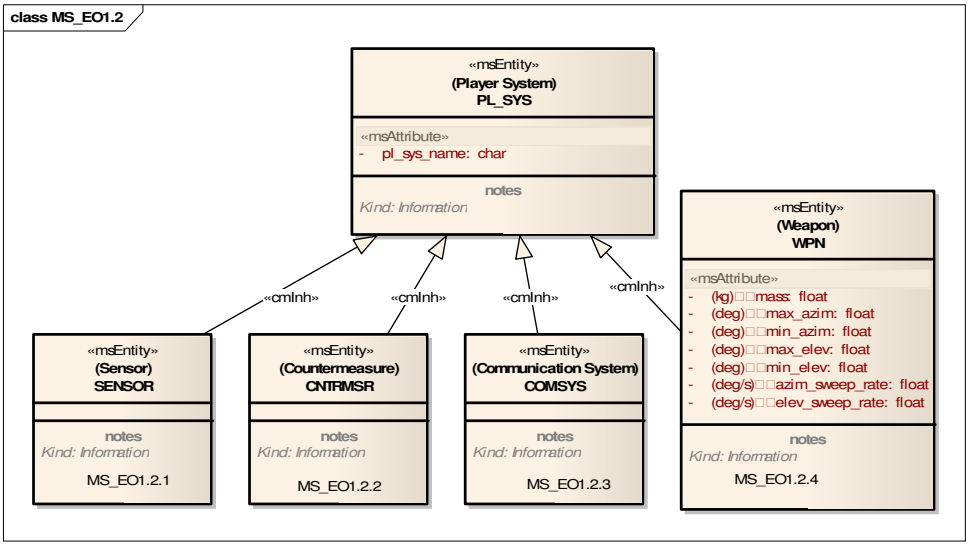


Figure 16: MS EO Diagram 1.2

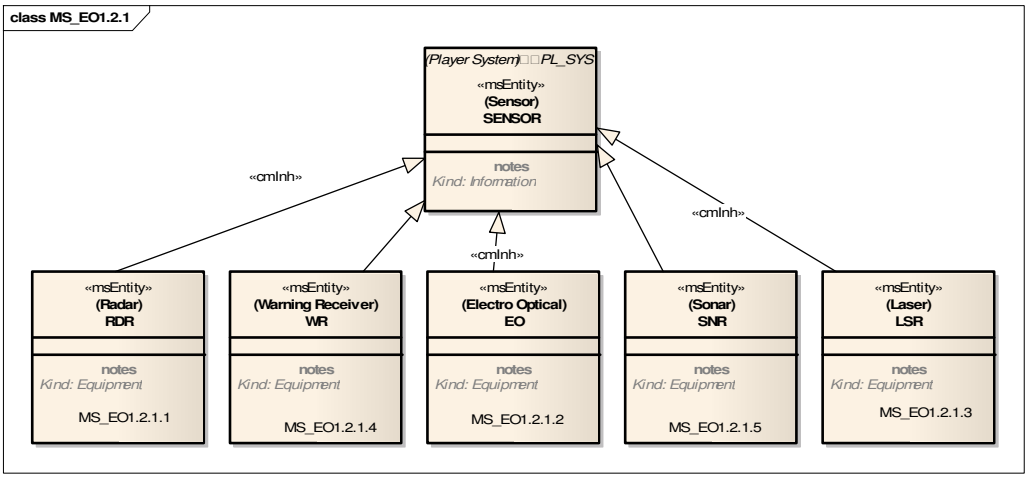


Figure 17: MS EO Diagram 1.2.1

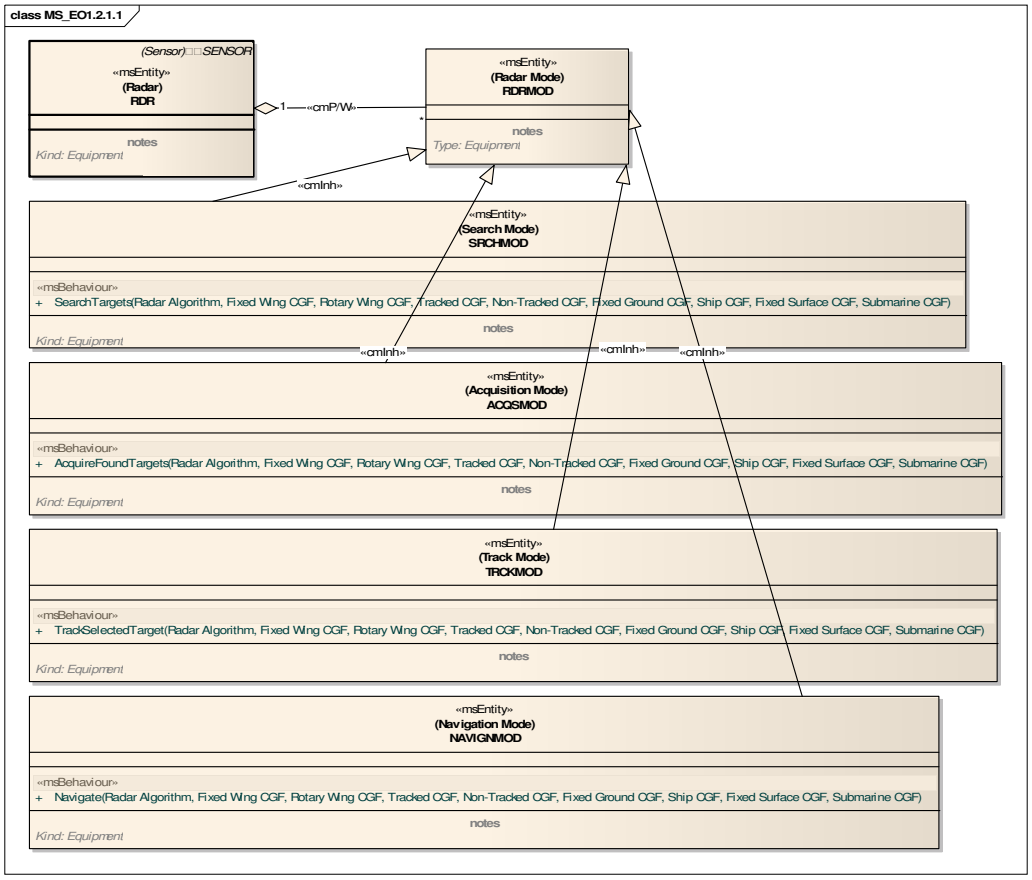


Figure 18: MS EO Diagram 1.2.1.1

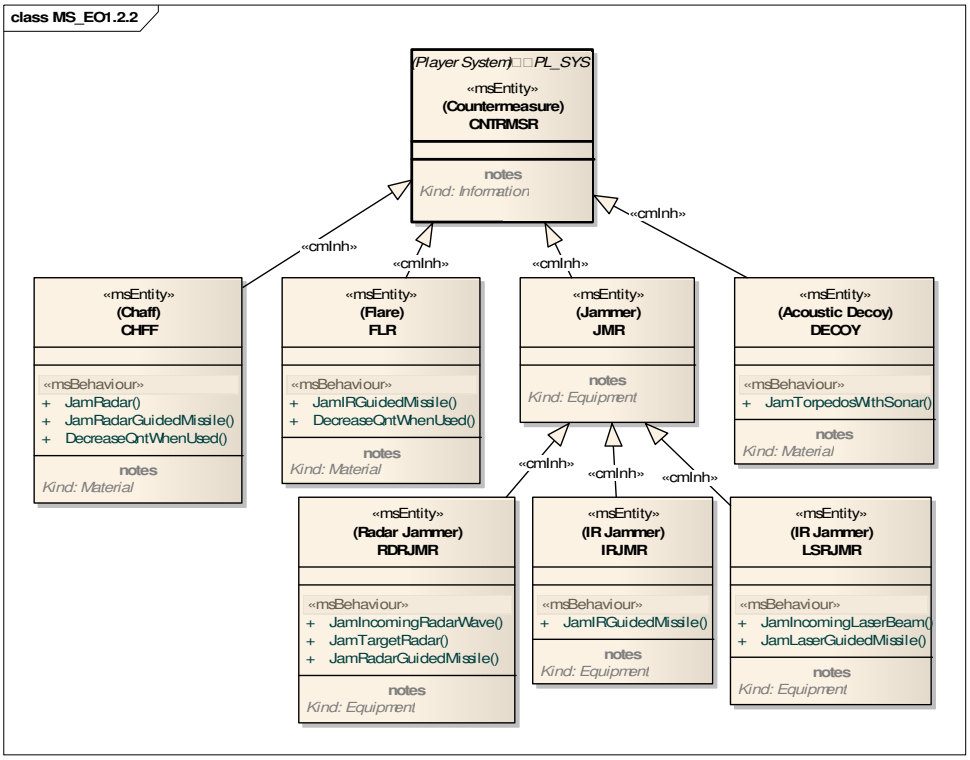


Figure 19: MS EO Diagram 1.2.2

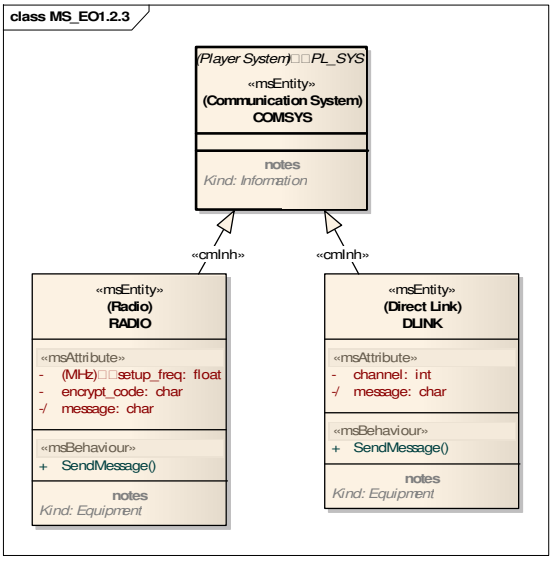


Figure 20: MS EO Diagram 1.2.3

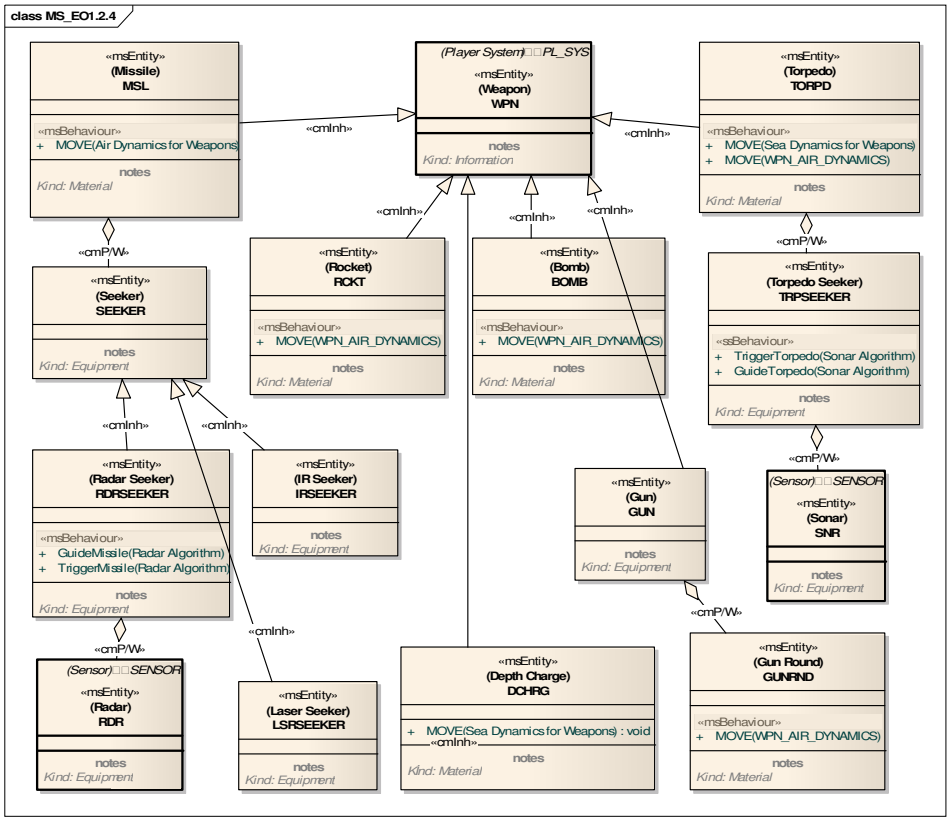


Figure 21: MS EO Diagram 1.2.4

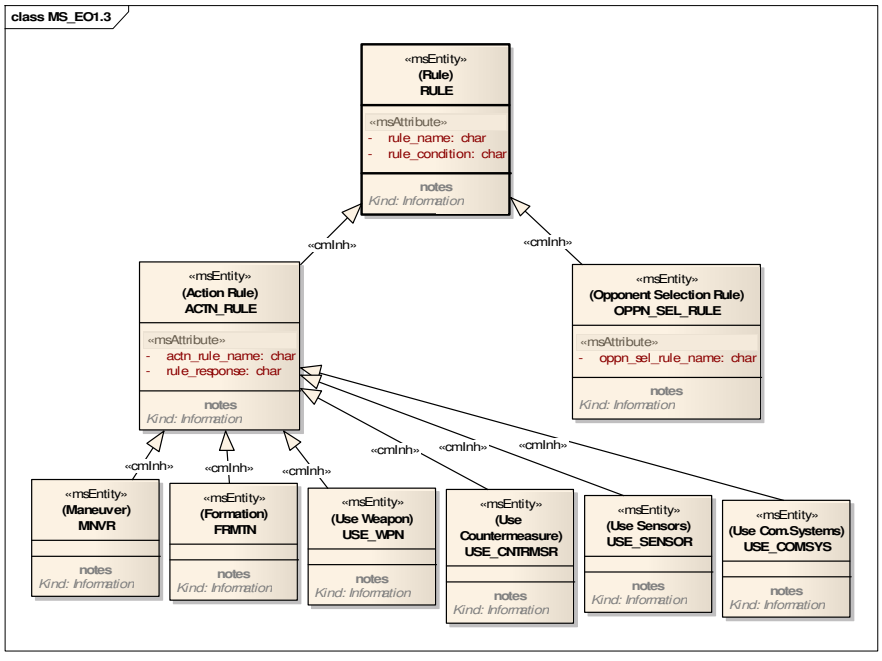


Figure 22: MS EO Diagram 1.3

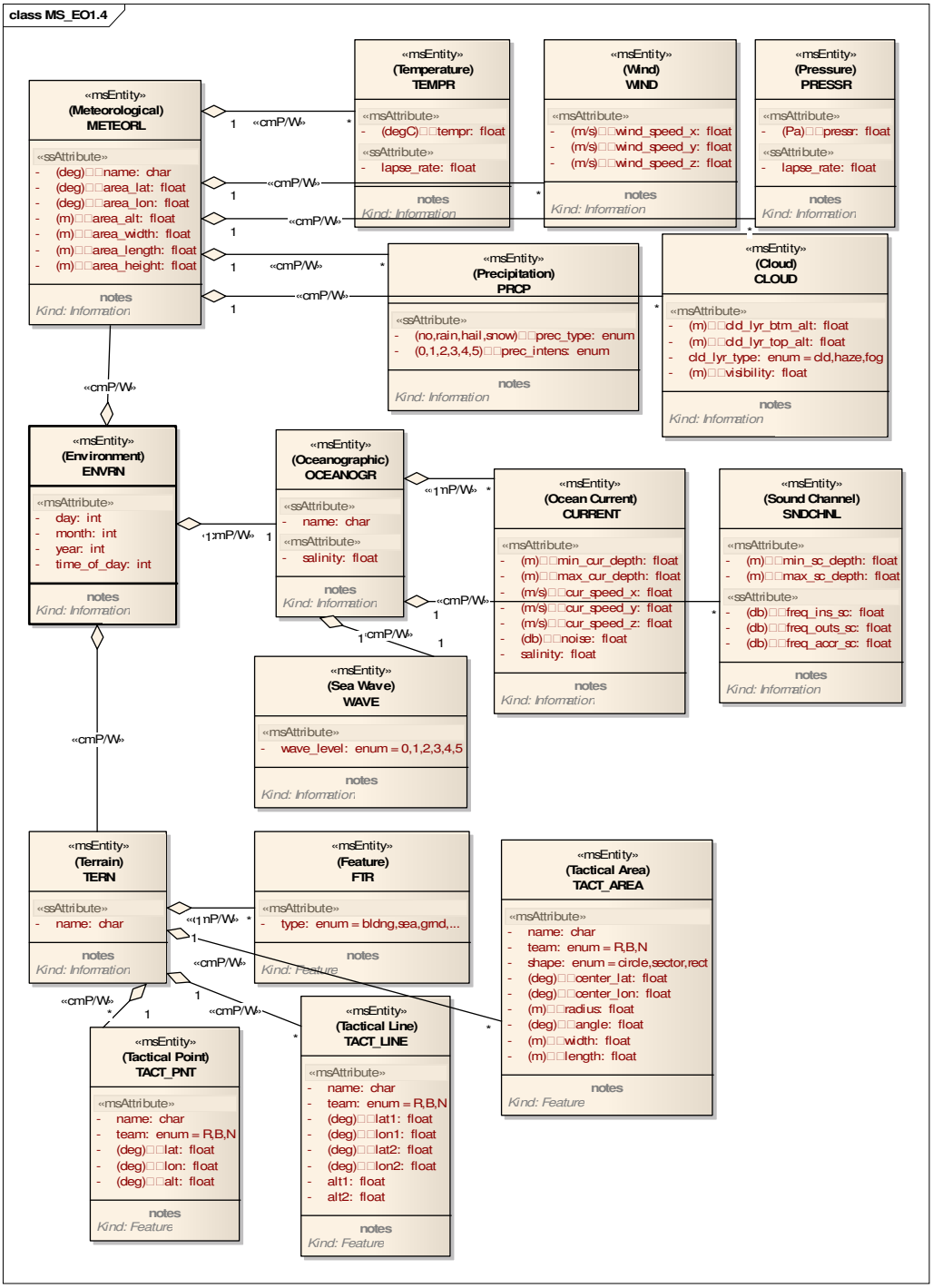


Figure 23: MS EO Diagram 1.4

4.4.3.2 Command Hierarchy Diagram

Command hierarchy diagram for SES is depicted in ***Figure 24: MS CH Diagram 1***. As the system mostly includes synthetic elements, there are not much elements in CH diagram. Captain, major and senior major actors are all the actors that have SS roles. The other actors depict different team types that can exist in battlefield. That team of actors is necessary to conduct various military operations on battlefield, which will be explained in MisSp and WF diagrams. Following the same color code, all actors are depicted in light pink as they are MS elements.

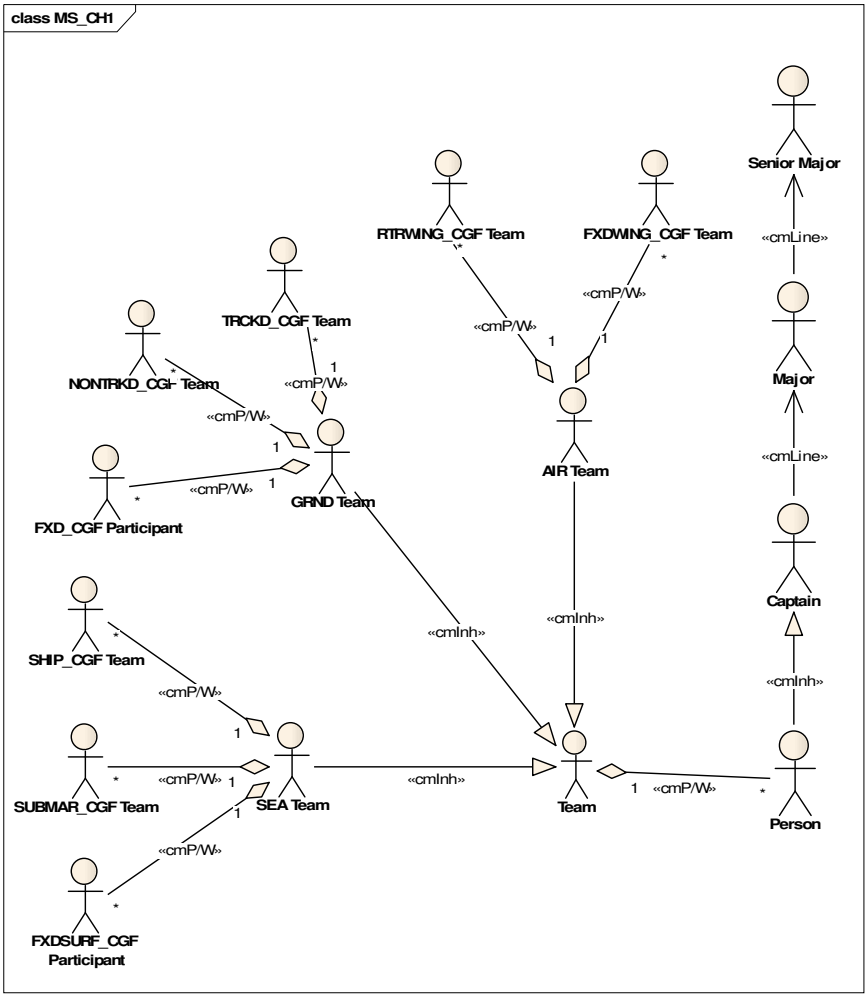


Figure 24: MS CH Diagram 1

4.4.3.3 Organization Structure Diagram

Organization Structure diagram for SES is depicted in ***Figure 25: MS OS Diagram 1***. The diagram includes elements referenced from other MS diagrams, indicating that they are actually entities specified in other diagrams. The relation between actors and player entities are explained in this diagram. This diagram is somehow different from a classical organization structure. This is due to the absence of real actors in the system, as the system simulates a synthetic environment. There are no roles defined for actors in synthetic environment. Here, this diagram is utilized to explain what entities (players) exist as team members in team actors.

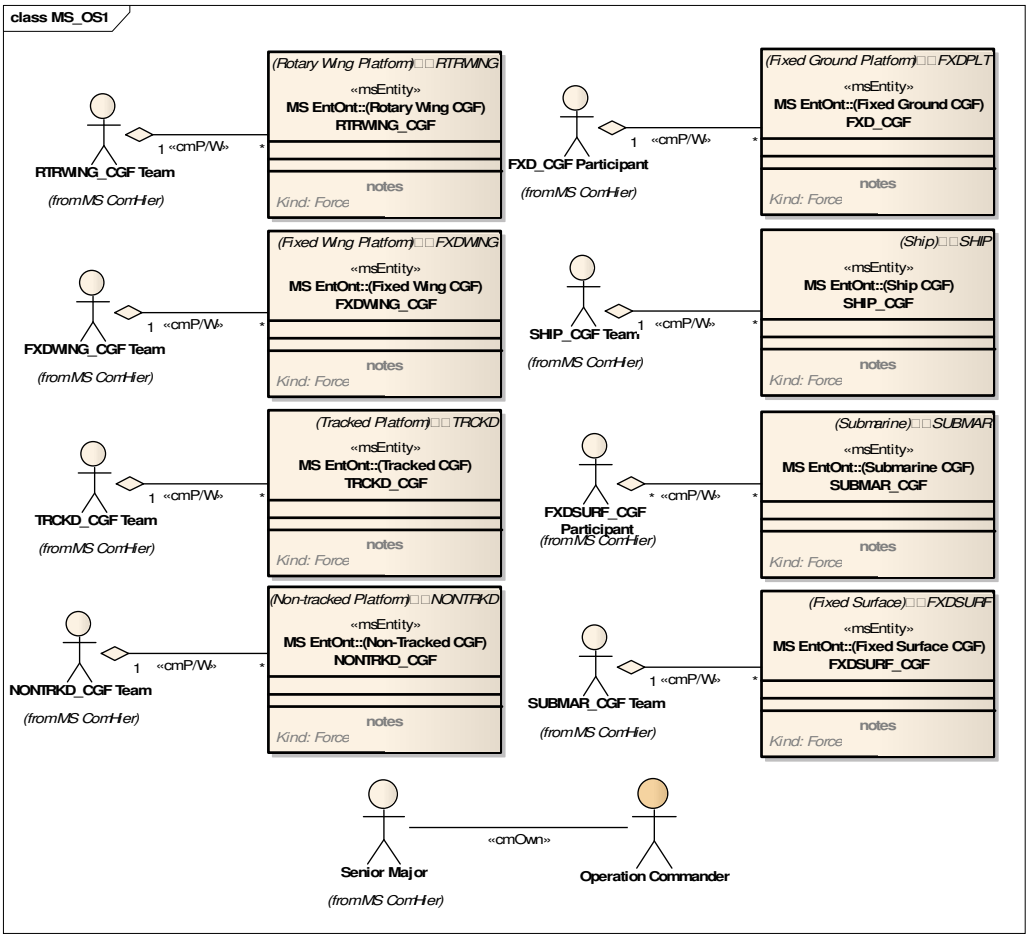


Figure 25: MS OS Diagram 1

4.4.3.4 Entity State Diagram

ES diagrams are developed for SES to define states and state transitions for different entities of system. The first diagram, ***Figure 26: MS ES Diagram 1***, explain states of RDR entity. There are many modes and events associated with transitions for RDR, as it is a complex player system entity. Notice that for all states, the name of entity that the state belongs to is identified within parentheses "()". There are mainly two kinds of states for RDR, first the one in which RDR makes emission (at left hand side), and the second in which RDR listens incoming waves (right hand side). There are three emitting modes that RDR can exist in. These are specified as separate RDRMOD entities that RDR has in EO diagrams. Due to various events under different situations that occur as explained on transition flow lines, RDR passes from one state to another, to update its behavior on target players. On right side, the state in which RDR is passively listening is depicted. This state (NAVIGNMOD) has sub states which occur when RDR detects some targets while listening. Those modes determine the level of detection for RDR. Looking at this diagram, one can understand the working principle of radar; what functions under different conditions it conducts to detect targets, what events can affect radar and how it works as a result of those events. The player on which RDR is assigned utilizes output data from RDR, which is not mentioned here, as it is not the aim of this diagram.

Rest of the ES diagrams is provided for LSR, EO, WR and SNR entities, that can be seen at [67]. Unlike RDR, LSR has two distinct states in which LSR conducts totally different functionalities. LSR uses one mode to determine the range of target, and uses the other one to designate the target. As the behaviors in two modes are not related to each other, there is no transition between those states. EO is a passive sensor and has similar modes to RDR's passive modes. EO waits for incoming light or heat (according to its type), and transitions between different levels of detection according to parameters it obtains during listening. The transitions occur due to events as shown in following diagram. The parameters that affect detection levels of EO are identified in detection algorithm of EO systems. WR is a passive sensor that has very similar states to EO. The difference is that, EO listens for incoming light or heat, whereas WR listens for incoming radar or laser waves. Lastly, SNR has a

similar structure to RDR, which has both active and passive modes. Due to events that occur, SNR switches between those states as depicted in the diagram.

ES diagrams play important role to model not only states of entities in military domain, but also events that affect those entities and make them change their behaviors. This is important to understand how entities work. For SE, states of player systems, which are affected from environmental conditions and adapt their internal workings accordingly, are depicted in ES diagrams.

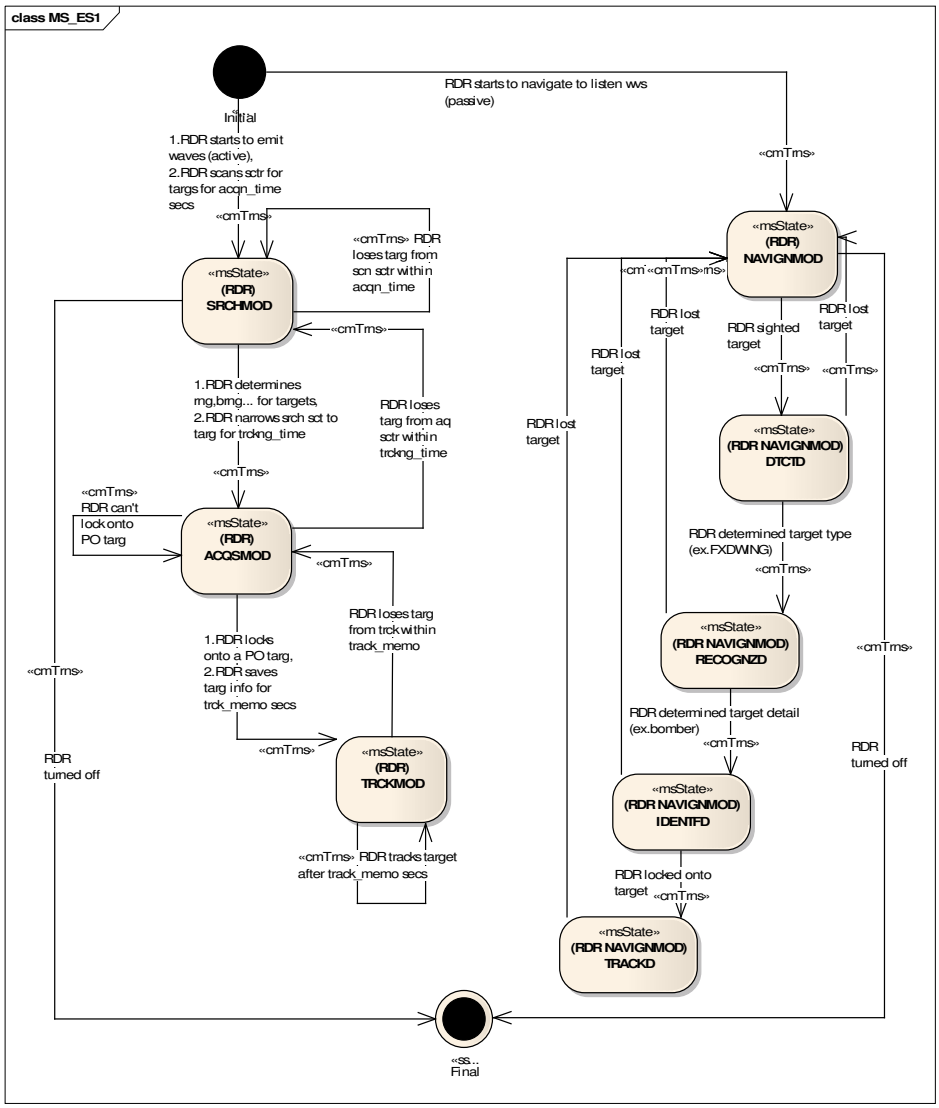


Figure 26: MS ES Diagram 1

4.4.3.5 Entity Relationships Diagram

Relationships between entities of system that was not depicted in EO diagrams are depicted here. ER diagrams may be used for many aims, depending on the structure of the system. In SES model, they are use to detail the behaviors of platform and player system entities. For this, algorithms are defined that specifies inputs and outputs of a behavior in detail, and usually utilized by more than one entity. In this way, looking at these diagrams, the reader understands how the entities behave, what aspects of the environment they are affected from and what is the expected fidelity of the simulation system. For example, if "Wind" is specified as an input in dynamics algorithm, and the algorithm is associated with a player; it is understood that the player dynamics considers the wind.

In ***Figure 27: MS ER Diagram 1***, rotary wing CGF, fixed wing CGF, fixed ground CGF, fixed surface CGF and fixed subsurface CGF entities are depicted. Airborne players are associated with their dynamics algorithms. Rather than a detailed description, it is important to demonstrate what the algorithm considers (inputs), what it calculates (outputs), and any basic information. For airborne players, many input attributes of different entities are specified. Additionally, environment entities that the algorithm considers are depicted as inputs. Outputs that are calculated by means of algorithm and the location attributes of player are specified. Looking at the definition for dynamics algorithm, the fidelity of calculations can be understood.

Collision algorithm is also depicted on this diagram, which is an algorithm used by all entities in the system. A short description of algorithm is provided, expressing that "when bounding volumes of two entities cross each other, entities are damaged". ER diagrams exist for also TRCKD_CGF, NONTRKD_CGF, SHIP_CGF and SUBMAR_CGF entities that can be found at [67]. Fuel consumption and maneuvering algorithms are defined for mobile entities. Rest of ER diagrams shown at [67] describes the algorithms used by sensors and the dynamics of weapons in air and sea environments. These algorithms explain how sensors detect targets, and what attributes they consider to detect. The environmental conditions they are affected from are also expressed as inputs.

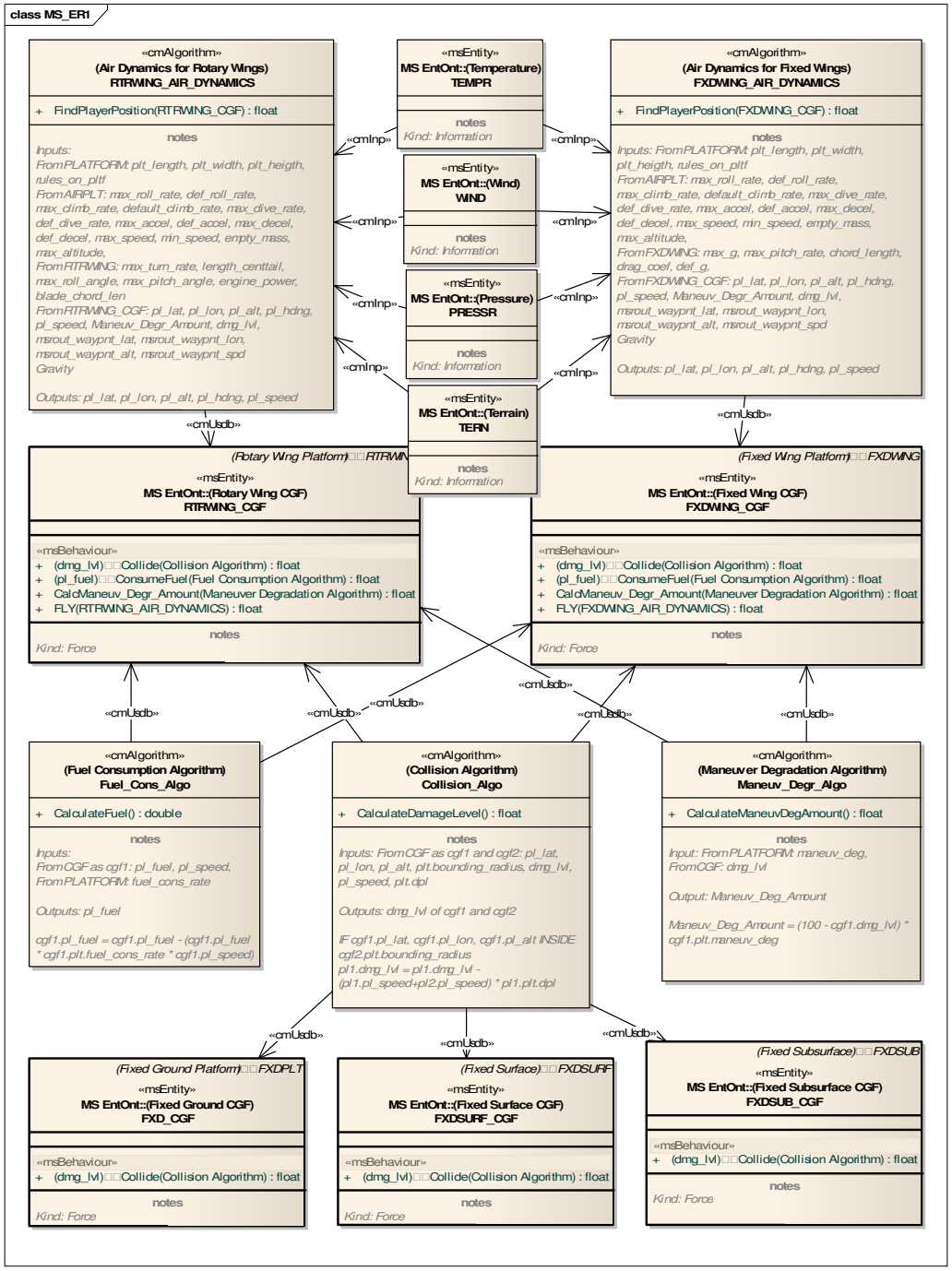


Figure 27: MS ER Diagram 1

4.4.3.6 Mission Space Diagram

Mission space diagrams are heart of conceptual modeling, providing basic missions to be executed. Mission space diagrams provide the basic information of what the objectives and the high level tasks of the system are, including actors, relations between missions, inputs and outputs. Mission space diagrams of SES explain the objectives of the system in high level. Having knowledge of all elements in the system, missions of the system can now be expressed in detail. Considering all information obtained until now, the system has missions to conduct various battlefield operations in which different types of players exist with different behaviors; and players use many types of player systems. In this way, the system aims to create infinitely many variations of different operational situations.

As it is not possible to model all variations of battlefield operations, generic operations to be conducted by system are summarized in MisSp and WF diagrams. The first MisSp diagram determines the highest level objectives of the system, as shown in ***Figure 28: MS MisSp Diagram 1***. The main mission of the system is to conduct operations to attack enemy units. This can include operations in which ground enemy units are attacked by air units, low level attack is conducted between rotary wing air units and enemy air units are attacked by ground units. Other than that, area surveillance can be conducted by airborne forces. All of the missions are associated with a terrain object, on which the related mission will be executed. Each mission has an objective, and a measure to evaluate if the objective is met. For example for first mission, if number of enemy ground units that can attack is zero, than the operation is successful. The actors that execute each mission are determined.

Details of mission 1 are provided as a separate MisSp diagram in ***Figure 29: MS MisSp Diagram 1.1***. This includes missions to be conducted by fixed wing and rotary wing forces on battlefield during a ground attack, including movements of forces, usage of sensors, attacking with weapons, avoiding enemy attacks and communications. Each of the missions has also objectives and measures, and actors executing them. The number of mission space diagrams and missions can be increased, according to how users intend to utilize the capabilities of the system.

Here, basic missions that are expected from the system are provided; and first mission is decomposed into lower level missions. The details of how to conduct each lower level mission go beyond the scope of MisSp diagrams. They will be explained in WF diagrams in detail in next section.

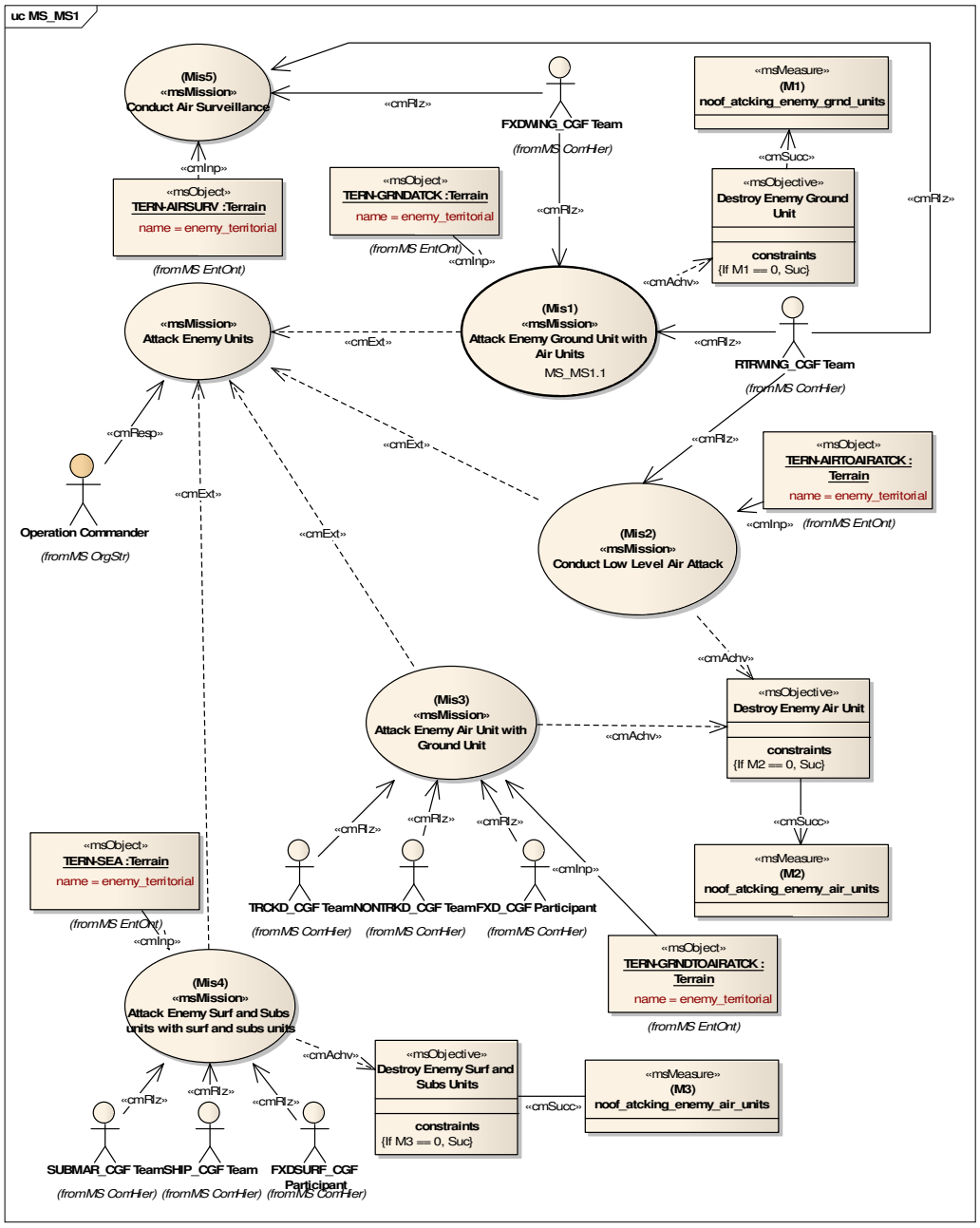


Figure 28: MS MisSp Diagram 1

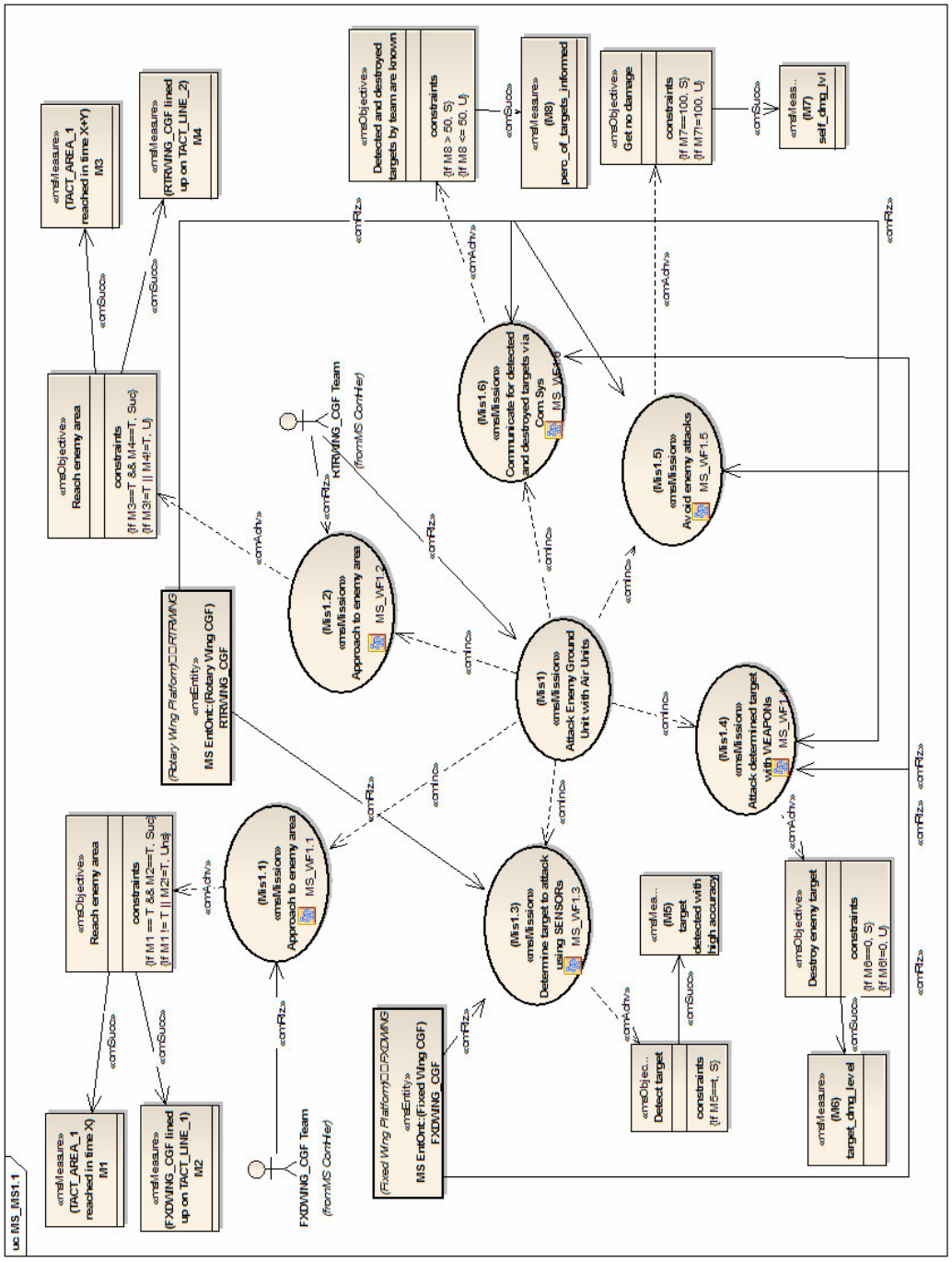


Figure 29: MS MisSp Diagram 1.1

4.4.3.7 Work Flow Diagram

Work flow diagrams aim to detail high level tasks that are specified by mission space diagrams. For each mission stated in MisSp diagrams, at least one WF diagram shall exist to describe the process to execute that mission. In SES model below, WF diagrams are provided for missions stated in ***Figure 29: MS MisSp Diagram 1.1***.

First WF diagram, ***Figure 30: MS WF Diagram 1.1***, explains how fixed wing forces go to operation area in the first place for an air-to-surface attack. Objects created from tactical line and point entities are used to depict tasks using those lines and points. Some of the tasks are conducted by team of fixed wing forces, whereas some are conducted by a single force. Second WF diagram, which is not placed here but can be found at [67], explains a similar mission for rotary wing forces. They reach operation area after fixed wing forces, and wait for them to complete their attack. The other WF diagrams explain the tasks conducted by forces to attack enemies under different conditions. In ***Figure 31: MS WF Diagram 1.3***, it is observed that while following mission route, forces continuously check sensors and tries to determine PO. When data from sensors are enough, PO is selected. Other activity to conduct on a selected PO is attacking PO with weapons, as described in ***Figure 32: MS WF Diagram 1.4***. After deciding to attack PO, force selects one of its weapons considering different conditions. For example, if target is a fixed SAM site, player cannot get close to it, so it has to attack that enemy with a radar guided missile. Other than attacking enemies, forces have to get precautions to avoid attacks of enemies. The diagram for this can be found at [67]. For example, when a radar guided missile attacks the force, it tries to jam it by releasing chaff. Lastly, forces communicate with each other during operation, for which WF diagram can be found at [67]. As they are all airborne players, they only use their radio for communication.

WF diagrams are utilized to describe many possible operations and activities that can be conducted in the system. For SES, how entities defined in previous diagrams are utilized in battlefield is depicted. If required, these diagrams can be more detailed to specify more complex activities.

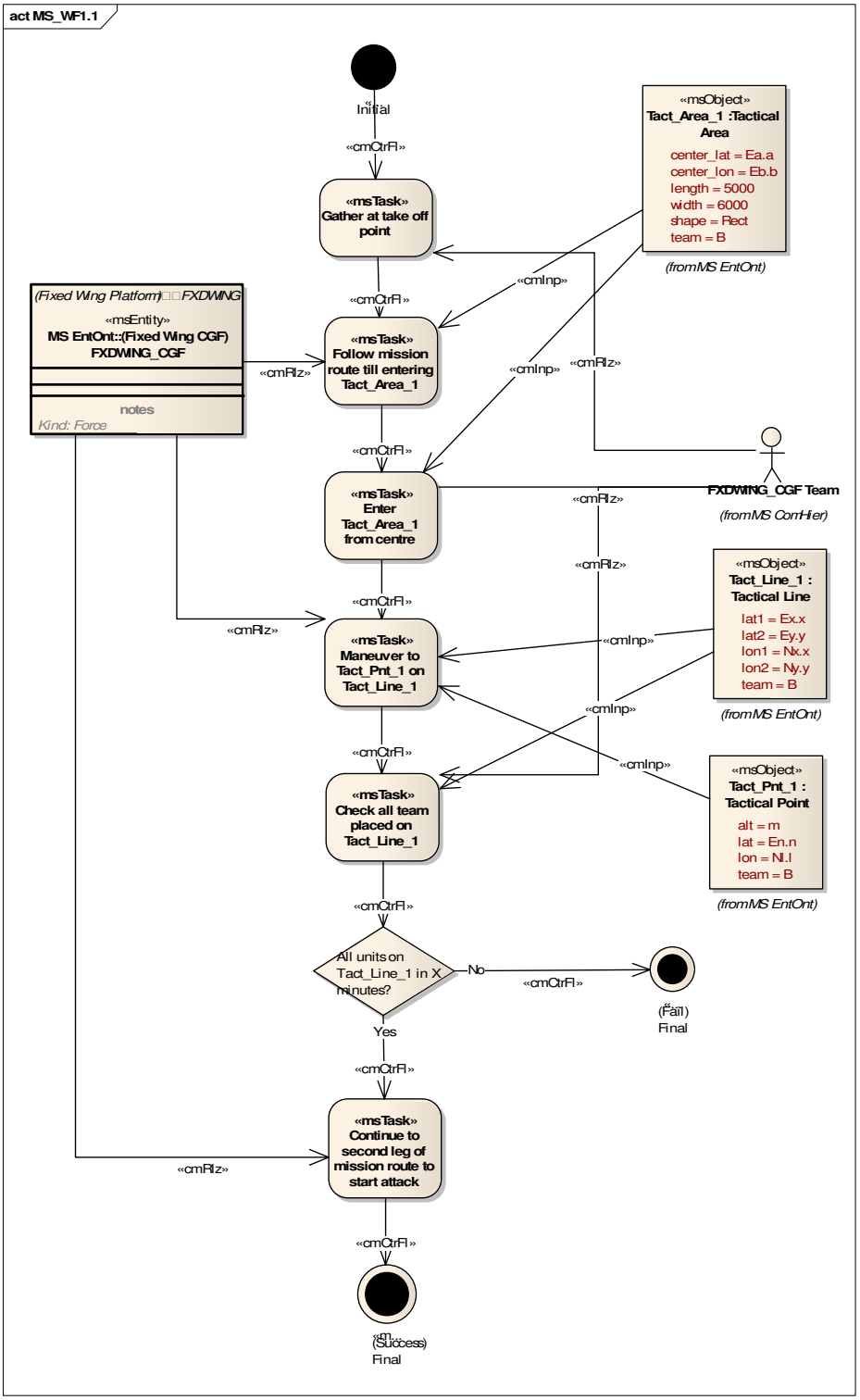


Figure 30: MS WF Diagram 1.1

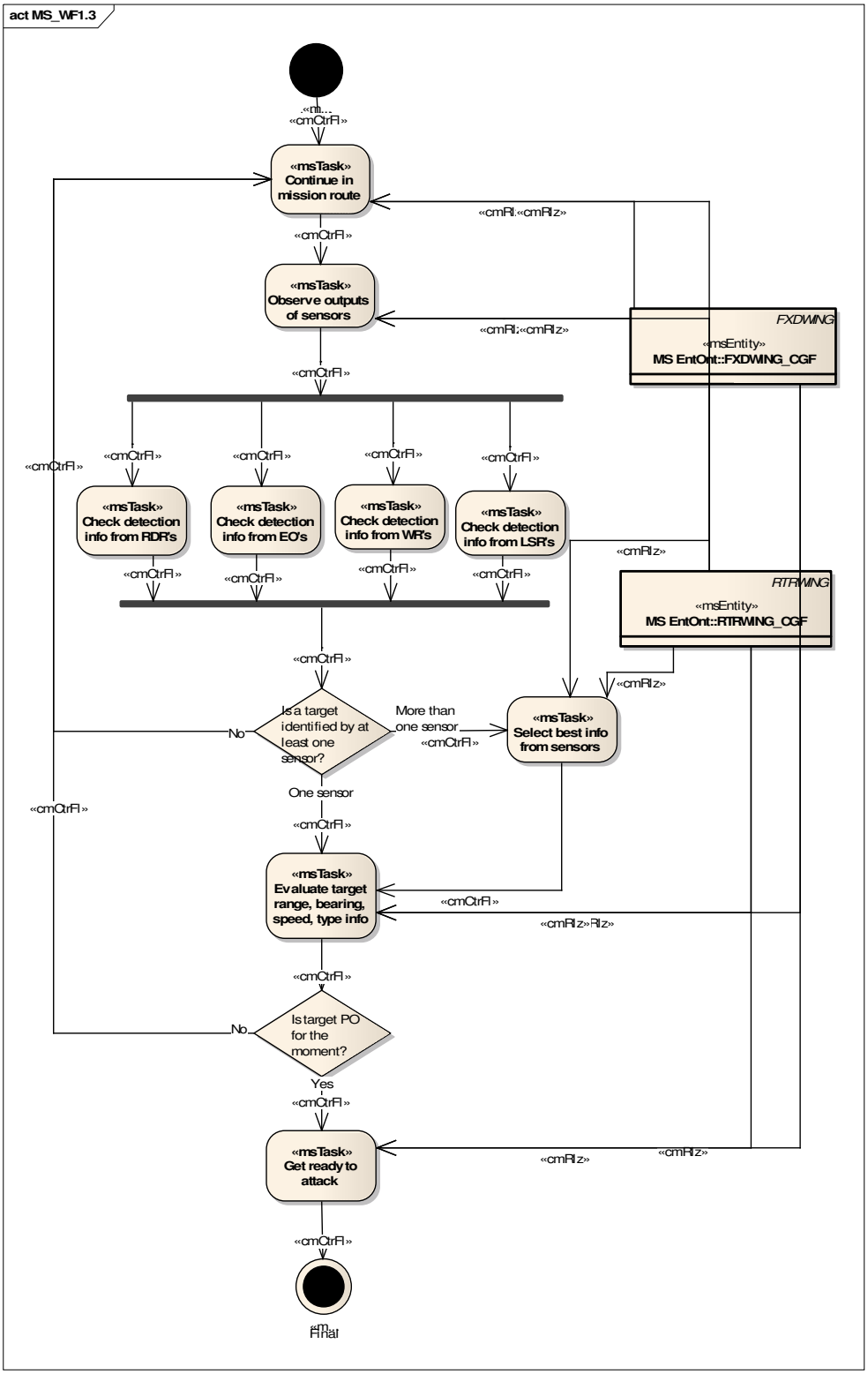


Figure 31: MS WF Diagram 1.3

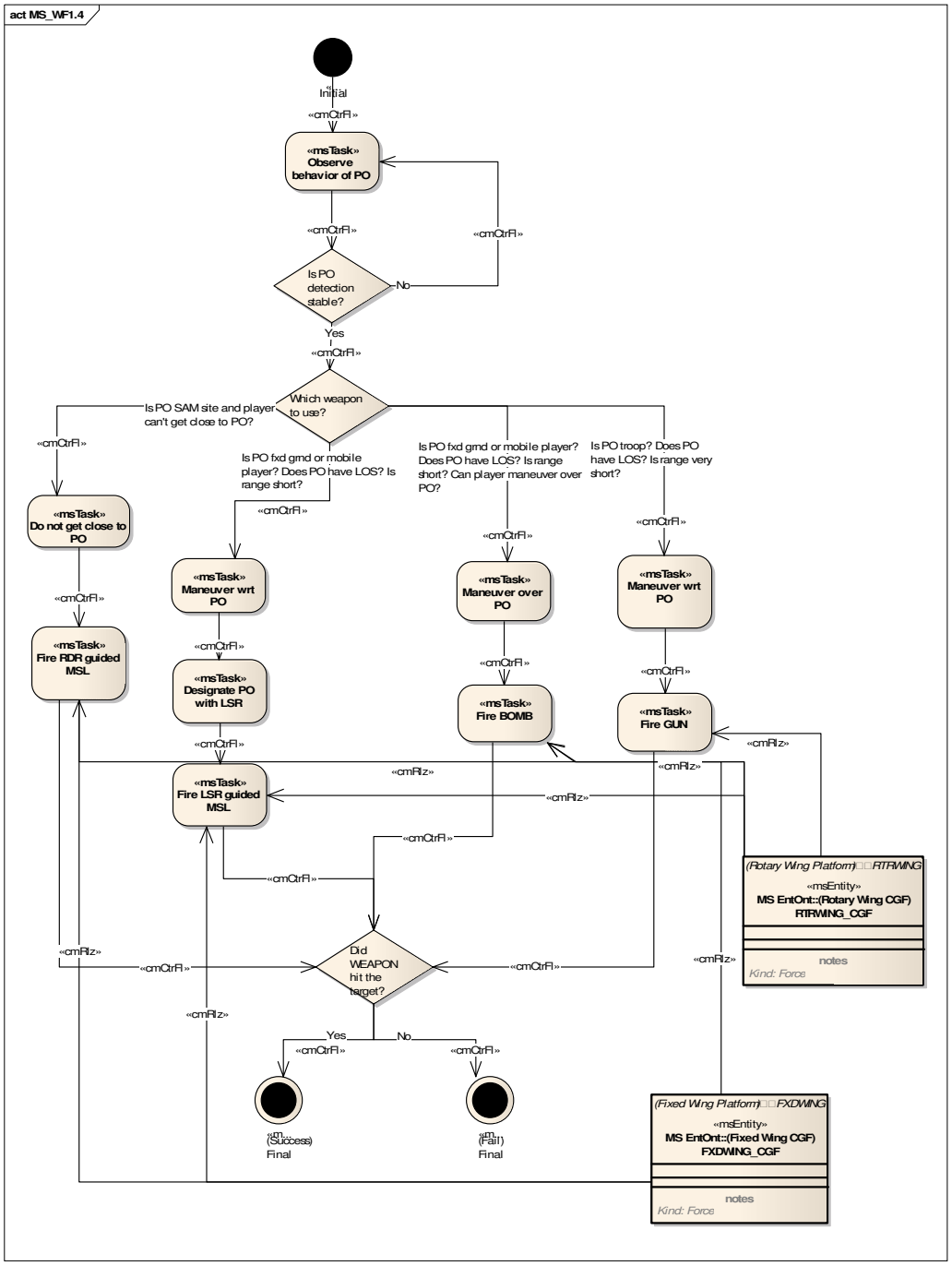


Figure 32: MS WF Diagram 1.4

4.4.4 Step 4 – Develop Simulation Space CM Diagrams

In this section, simulation space conceptual model diagrams of SES project will be developed, by using the methodology described in ***3.6 Step 4 – Develop Simulation Space CM Diagrams***. Model elements identified in section ***4.4.2 Step 2 – Identify Model Elements***, and newly added elements are used in diagrams. All types of diagrams are provided in below sections that compose CM for SS of SES.

4.4.4.1 Entity Ontology Diagram

In ***Figure 33: SS EO Diagram***, EO diagram for SS is shown. This diagram depicts the basic components of the simulation system, and inheritance and part/whole relations between them. In this way, physical structure of the simulation system is grasped. If the entity is hardware like a station, or a mixture of hardware and software, "Kind" is specified as "Facility". For software entities, "Kind" is specified as "Software". In this manner, hardware and software components are differentiated. In this diagram, SES is not in the highest level of part/whole relations, it is in the second level. This is because the diagram also describes the higher level system in which SES resides. This provides an understanding of the whole system and objectives. As the color coding indicates, all entities are SS elements in light green.

EO diagrams for SS are important in conceptual modeling. In the initial phases of SDLC, mostly, users and developers cannot come to an agreement just because they couldn't clarify the physical structure and software components of the system and relations between them. Although a basic depiction of simulation system entities are enough, as done in EO diagrams, this solves many confliction between users and developers.

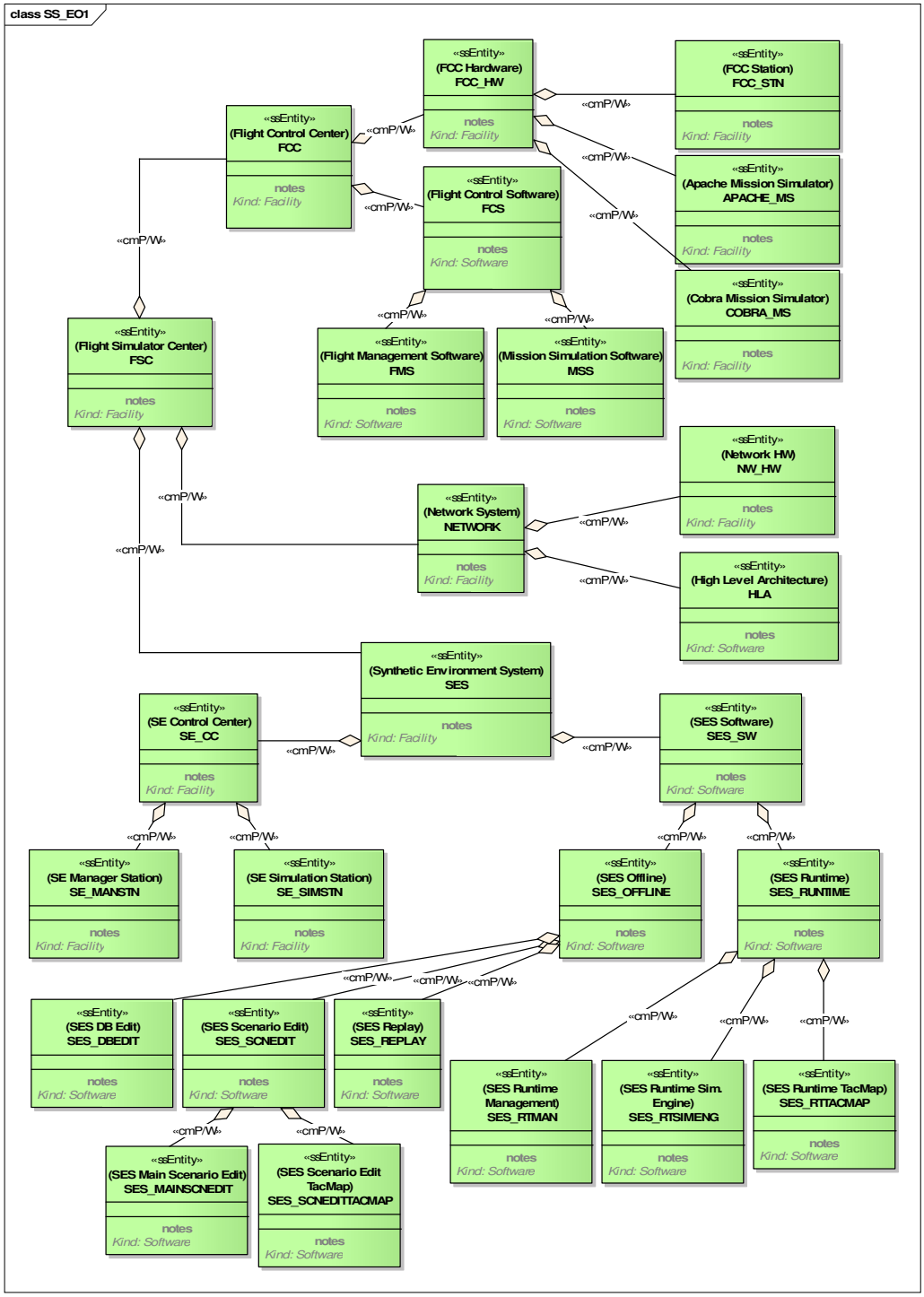


Figure 33: SS EO Diagram

4.4.4.2 Organization Structure Diagram

In ***Figure 34: SS OS Diagram***, roles that execute activities in simulation system are depicted. As there are not any actors specific to simulation system, all actors were defined in mission space CH diagrams. In this diagram, actors from that diagram are utilized. They are just references from related MS diagram, meaning that they actually do not exist on this diagram, but is placed here as a reference to show relation of them to elements existing in this diagram. It is an exception that MS elements exist in SS diagram.

The roles defined in SS OS diagram are the ones that execute tasks in WF diagrams. The operations they conduct in system will be explained in those diagrams. For all roles, an attribute specifying that they have M&S background is defined. SS roles are depicted in dark green color.

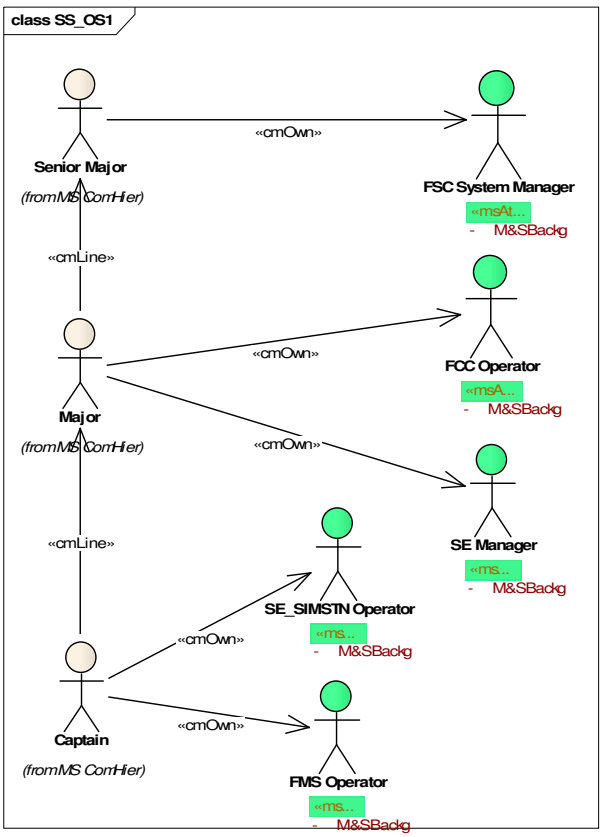


Figure 34: SS OS Diagram

4.4.4.3 Entity State Diagram

ES diagrams in simulation space conceptual modeling depict the states that the simulation system might be in, and transitions between them. ***Figure 35: SS ES Diagram*** shows that the simulation system has two modes in which different components of the system run. States for SES are shown in light green, as they are SS elements. The events that trigger transition between states are shown on transition lines. This diagram shows the states of SES system as a whole. Of necessary, states of smaller parts of the system, like a software or hardware component, can also be depicted in these diagrams.

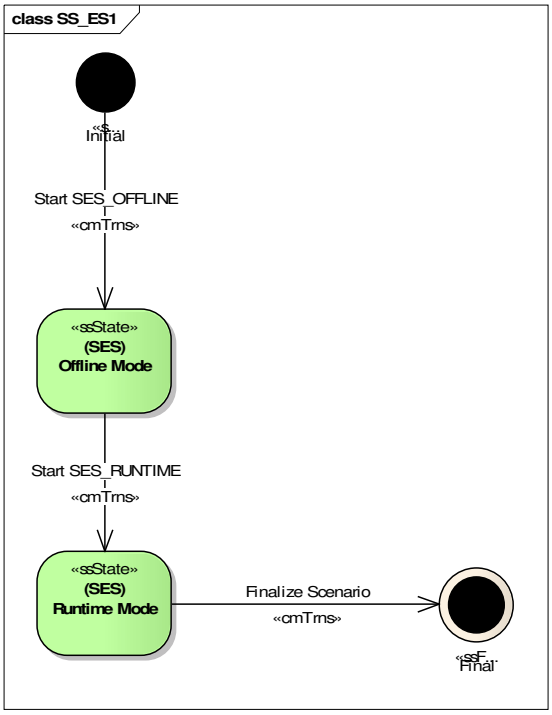


Figure 35: SS ES Diagram

4.4.4.4 Entity Relationships Diagram

ER diagrams are very important for SS conceptual modeling, as they can be used to model many aspects of system, by means of their flexible structure. In this model, this diagram type is utilized to more clarify relations between components of simulation system. In ***Figure 36: SS ER Diagram 1***, for each software entity, where that entity runs on is specified. Each software entity runs on one or more facility entities, which are hardware stations. Other than that, it is also important to specify which software entities are running on different states of system. To depict all this information, all software entities are listed in the middle of diagram, with states at left hand side, and facilities at right hand side. For related entities, generic "cmRln" is utilized, with "Run On" expression on them, explaining the relation.

Not only SES entities, but also higher level entities are included to increase the understanding of the system. For example, it is shown that HLA software runs on Network HW, and it runs when SES is in "Runtime Mode" state.

In ***Figure 37: SS ER Diagram 2***, the relation between SES, network and FCS are explained in more detail. It can be observed that there is no direct relation between SES_SW and FCS, all interaction is provided by means of HLA.

Notice that these diagrams are used to depict relations between entities that in fact exist in other diagrams. SS entities on these diagrams are references from other diagrams. ER diagrams are complementary to EO diagrams. The physical structure of the system that is introduced in EO diagram is explained in more detail in ER diagrams. In this way, readers obtain a wider idea on how the simulation system works.

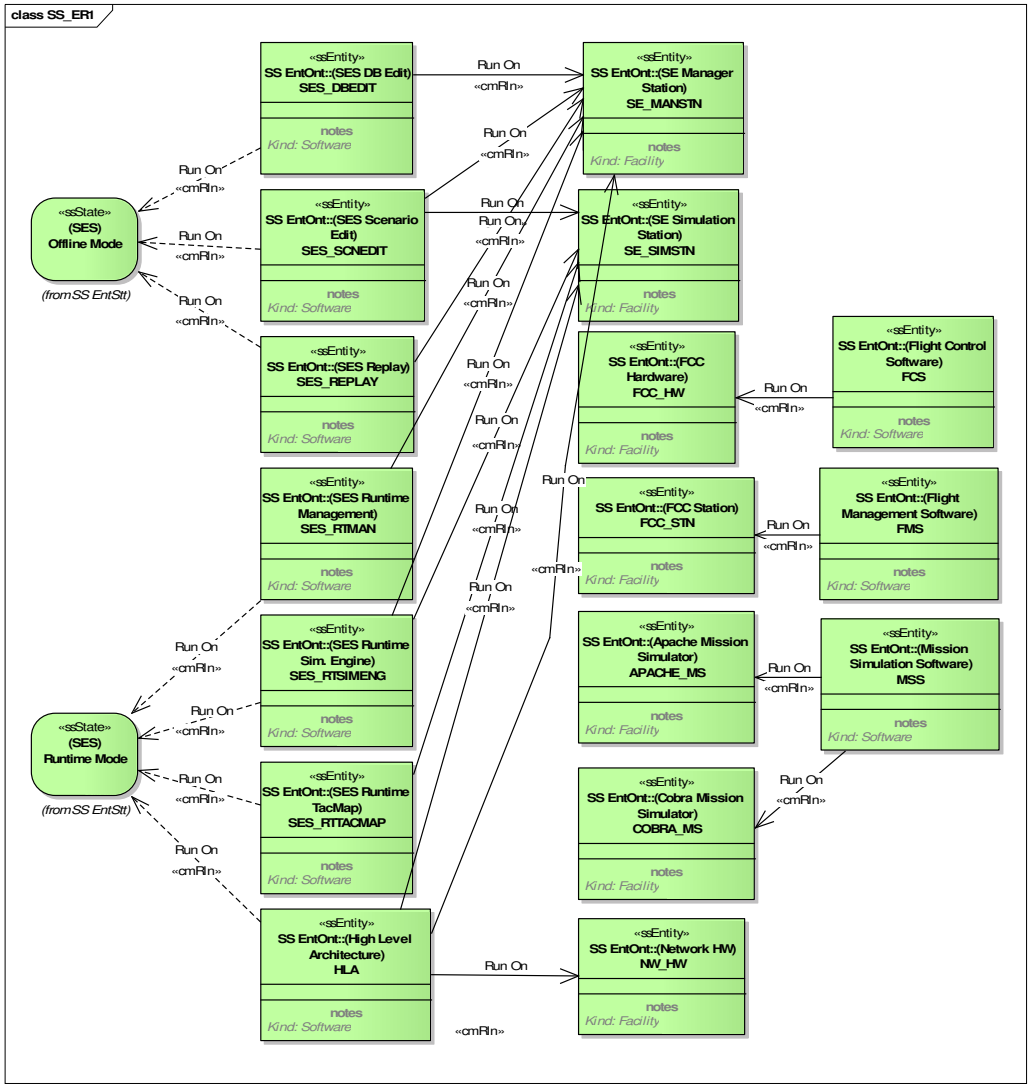


Figure 36: SS ER Diagram 1

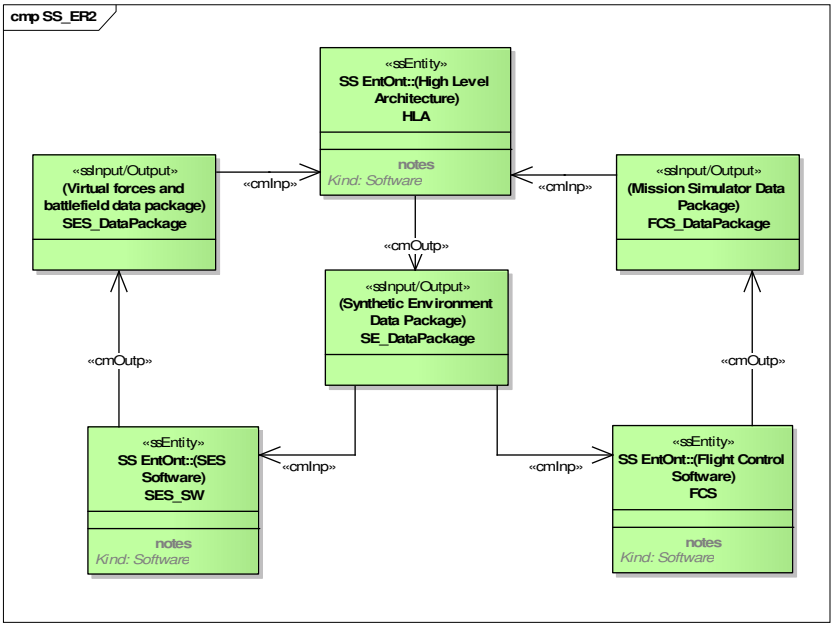


Figure 37: SS ER Diagram 2

4.4.4.5 Work Flow Diagram

WF diagrams for SS describe the execution order of activities and activities that the external users of the system, depicted as roles, execute. Inputs and outputs used in tasks are included. The order of tasks is depicted in a timely manner, considering detailed conditions, parallel executions etc.

WF diagrams are modeled in a hierarchical manner, which link to more detailed diagrams. For example, first diagram (***Figure 38***) is the most general WF diagram that depicts general tasks in the system. On this diagram, links on four main tasks are provided. Using these links, one can reach other four diagrams that explain how these activities are conducted in detail. Only the task with the link to SS_WF1.3 diagram is placed as an example in this study. Six more diagrams with different hierarchical levels can be seen at [67].

In these diagrams, all the tasks that are conducted to run the simulation system, including the activities needed to start system, generate records, manage an ongoing

operation, and evaluate the completed operation are included. Within this context, data definitions operations are also explained.

As all the tasks are SS activities, all entities are shown in light green color. Inputs and outputs in the system are also depicted with input and output relations to tasks, and they are also light green. On each task element, the related software entity of that task is written. In this way, one can understand on which software entity the activity is conducted.

When a decision is required to select a task to conduct within more than one option, decision points are used. When necessary, loops are indicated. For example in diagram ***Figure 39: SS WF Diagram 1.3***, after decision "Control Scenario", if user implements the task "Terminate Scenario", the flow goes to the beginning. If, when and while conditionals are also exploited when necessary. In the same example, these conditionals are used to reflect different behaviors of the system. In the same diagram, it is seen that runtime controls of the system are explained, each as separate tasks to be executed under different conditions, as stated under decision points. Runtime controls are scenario controls and player controls. Replay controls are explained in SS WF Diagram 1.4, which can be seen at [67]. Subtasks are generated on diagrams by placing them inside general task. In this way, tasks are defined better.

When there is an input from role of an actor while the actor executes a task, the type of that input is shown as a constraint on "realization" relation. Example of that is "Input: Via Keyboard" constraints on realization relations that exist in most diagrams. In this way, the interfaces that the users utilize to conduct operations on the system are identified.

WF diagram is the diagram type that provides the most detailed information on operation of simulation system. By means of it, the activities required to use functionality provided by simulation system, the timeline of those activities, the activities conducted by each actor, the conditions to conduct each task are explained. In this way, functionalities of the simulation system are clearly identified.

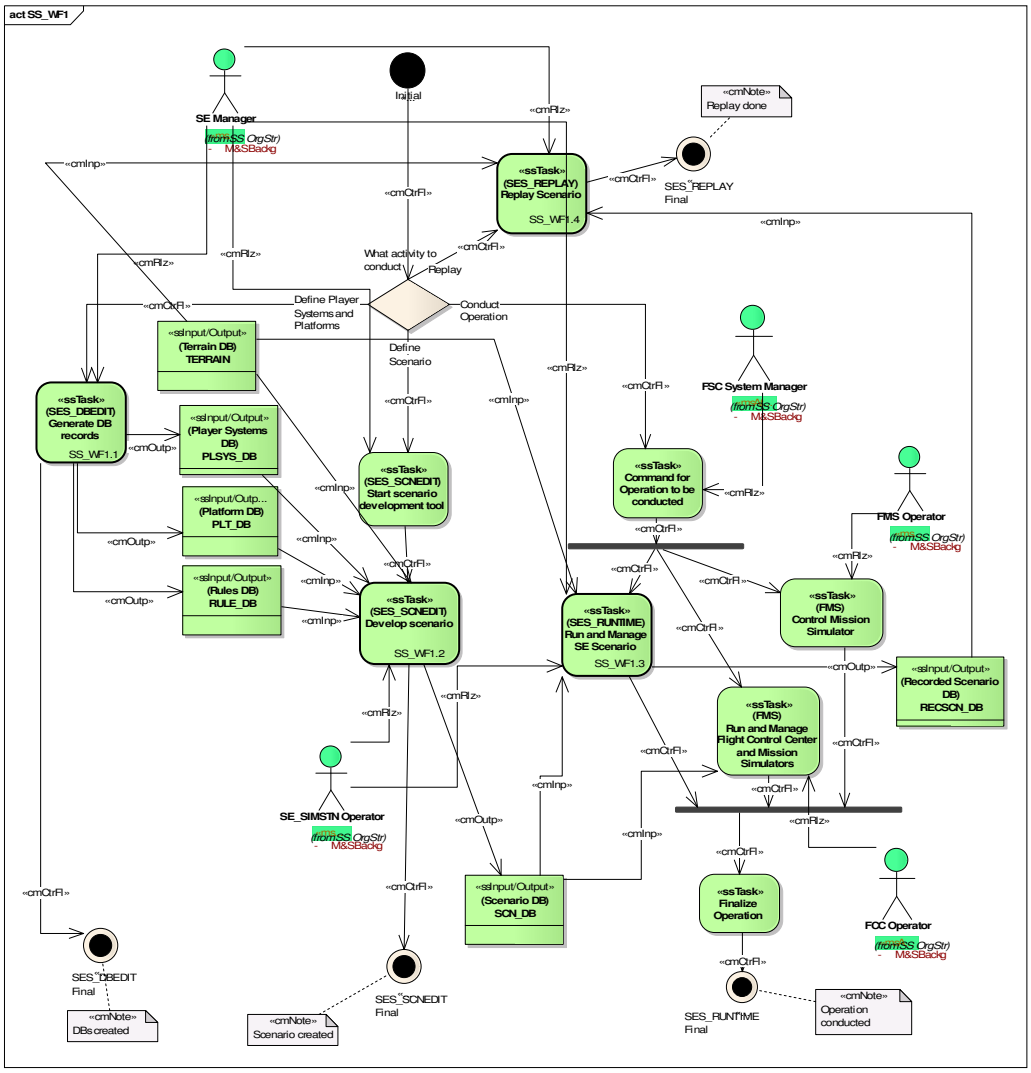


Figure 38: SS WF Diagram 1

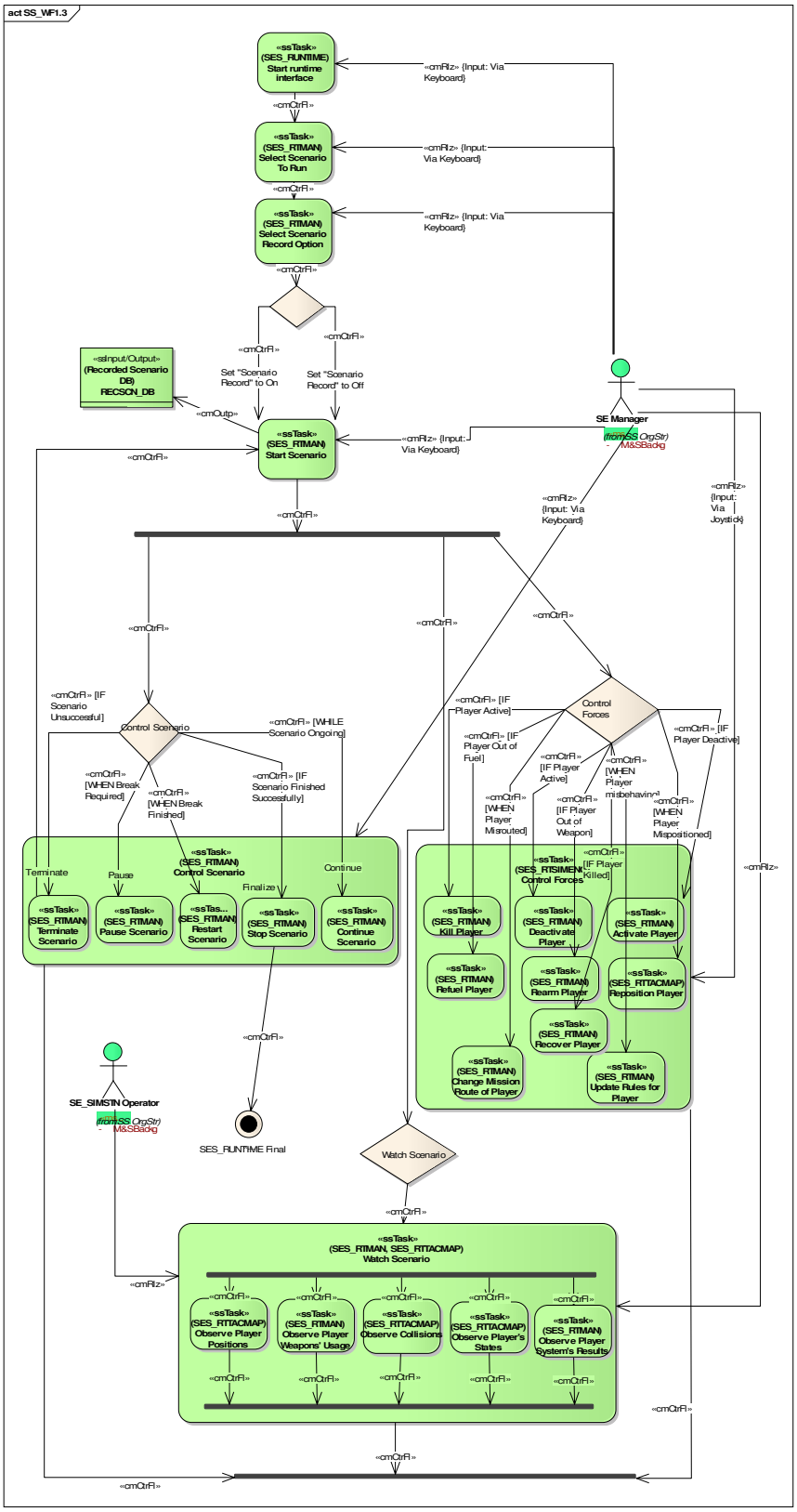


Figure 39: SS WF Diagram 1.3

4.4.5 Step 6 – Verify, Validate and Finalize CM

After developing all MS and SS diagrams and finishing CM, the next step is finalizing CM. CM needs to be verified and validated before adding CM to common repository. As explained in ***3.7 Step 6 – Verify, Validate and Finalize CM***, verification of CM shall be conducted with respect to syntax and semantics. Most of the syntactic and semantic validation is conducted automatically by means of CM development tool KAMA aims to provide, during CM development and after CM development by applying a process on CM. In that way, it is assured that all constraints of CM language are met in CM. In the case study research developed in this study, a third party tool is utilized, as KAMA CM development tool was not available yet. Syntactic and semantic validation is assured by means of defining profiles in that tool and by manual checks. If KAMA tool was used, a complete syntactic validation could be achieved. No systematic validation for the developed CM is conducted, although unofficial reviews are carried out by different experts to evaluate completeness and consistency of the model.

4.4.6 Step 7 – Develop High Level Design

Once development of CM is completed, it is time to move to design activities in simulation development life cycle. As an outstanding issue of this study, a guide to develop high level design by using CM as input is described in ***3.8 Step 7 – Develop High Level Design***. In this section, high level design for SES project will be developed and reported using CM as explained in previous sections, following the provided guideline. In following sections, each UML design diagram that can be developed using CM will be handled one by one, and developed diagrams will be presented and explained.

4.4.6.1 Class Diagrams and Package Diagrams

As described in section ***3.8.1***, class diagrams depict main parts of system and relations between them. Mission Space Entity Ontology, Entity Relationship and Simulation Space Entity Ontology diagrams are utilized to develop class and package diagrams. Some of the class diagrams of the system are introduced here.

First, SS EO diagram is used to specify packages that hold classes inside. Entities with kind "software" shown in ***Figure 33: SS EO Diagram*** are transformed into packages in a hierarchical manner indicated by p/w relations in EO diagram. The class diagram depicting only packages in the system is shown in ***Figure 40: HLD Class Diagram 1***. There is a big package for SES software, which includes main classes and all other packages for software components. Other packages for components and sub-components are indicated hierarchically. There is also HLA package to place classes that the system will implement as network software. These packages are expected to be implemented by developers in detailed design activities, except two of them. The classes of SES_DBEDIT and SES_RTSIMENG packages are shown in ***Figure 41: HLD Class Diagram 2***. In this diagram, entities in MS EO diagrams are utilized as classes. Each entity in MS EO diagrams is either a class in SES_DBEDIT or SES_RTSIMENG. Entities with kind "information" reside in SES_DBEDIT package, because they are classes to be defined by means of DB edit activities. Entities with kind "force, material or equipment" reside in SES_RTSIMENG package, because they are real entities that are simulated at runtime. Inheritance relations are transformed into generalization and part/whole relations to aggregation on class diagrams. In this way, main class structure of these two packages has been formed. Neither all entities in MS EO diagrams are placed nor are all attributes and behaviors of existing classes shown in this class diagram, because of space constraints of a page. Rather, by placing example applications, author aimed to increase understandability of diagram. In class "PLTF", attributes are shown. All attributes existing on MS EO diagrams are placed including initial values and enumerations, removing MS/SS classification. The units stated for attributes and indication of fixed/variable attributes are removed, but these shall be reported in design documents. Behaviors of some classes in SES_RTISIMENG package are shown. They are also used in the same way with CM diagrams. Quantities on p/w relations are transformed to multiplicities on aggregations. The rest of entities in MS EO diagrams shall be placed in this diagram with their attributes and operations, to complete this diagram.

The other element in class diagrams is interfaces. Algorithms in MS ER diagrams are transformed to interfaces. Inputs determined for the algorithm are listed as attributes

of interface. Interfaces have no behaviors. The diagram depicting interfaces and their relations in the system can be found at [67], on which interfaces that are used by different types of players are placed. Interfaces are associated with classes that use them. To decrease complexity, only attributes of two interfaces are shown and the rest are not shown. In CM diagrams, there are also entities which are input to algorithms. They are connected to interfaces with associations, and roles of them are specified as “used by”, as they are utilized as inputs by interfaces. Other class diagrams for the rest of the entities can be formed in the same manner. For example, class diagram for player systems and interfaces of them can be formed. In this way, class diagrams are formed as high level design of the system.

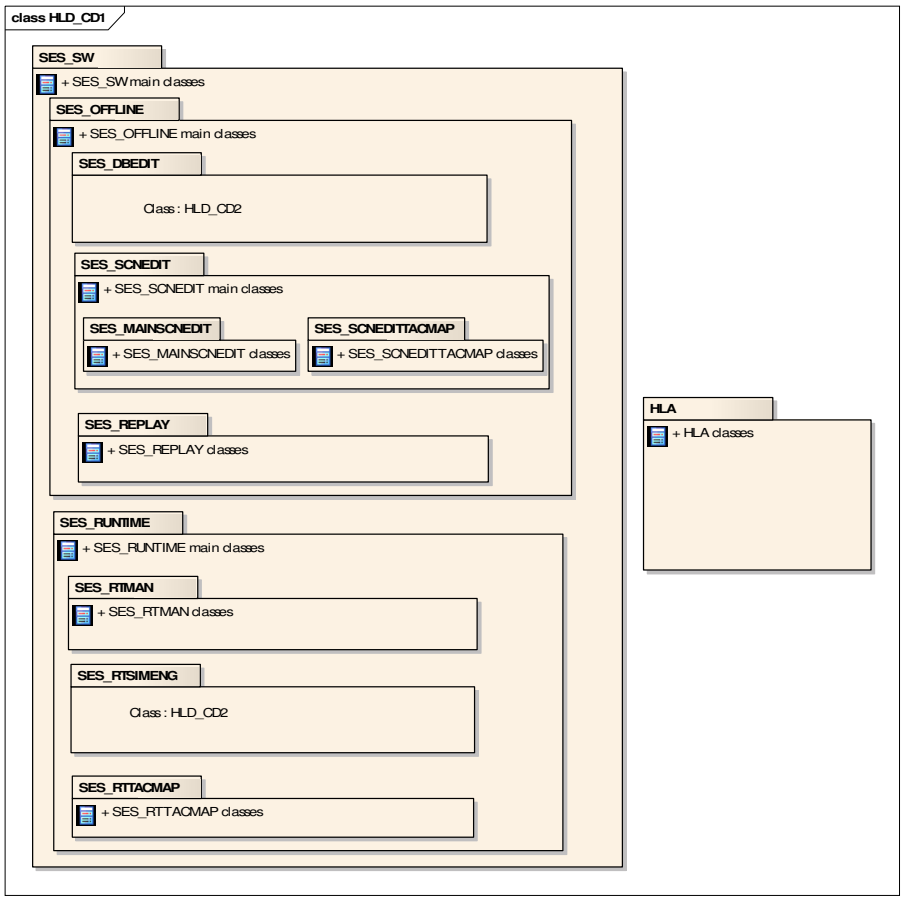


Figure 40: HLD Class Diagram 1

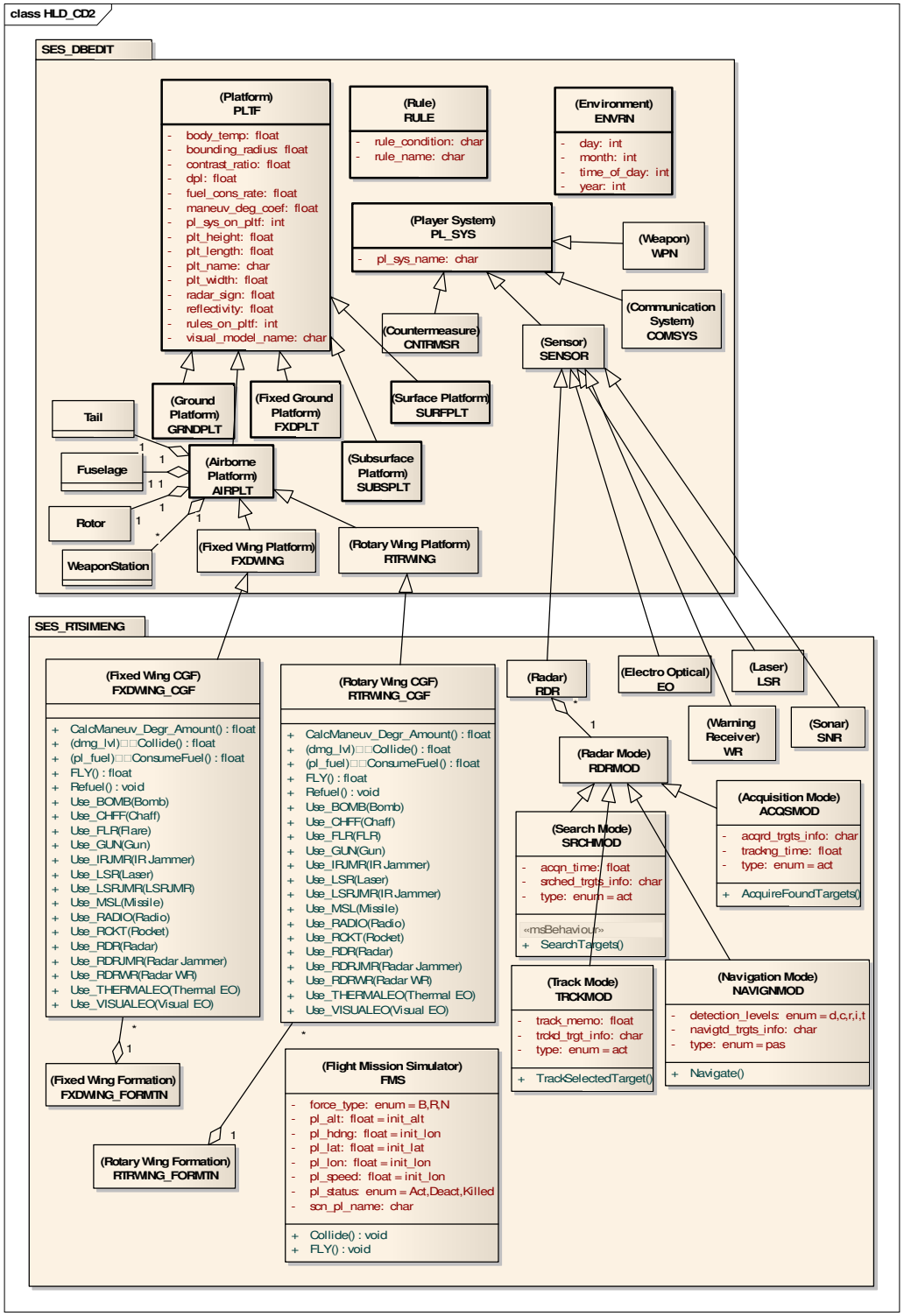


Figure 41: HLD Class Diagram 2

4.4.6.2 Object Diagrams

Object diagrams are a special case of class diagrams, in which instances of previously defined classes are introduced. In conceptual model of SES, object elements are used to determine specific instances of entities required by user, and values of attributes for them. They are placed in MS EO diagrams. They will be transformed to objects in UML object diagrams in the same way. As they are objects of different entities of kind "force", they don't have a direct relation between each other. UML Object diagram for SES can be found at [67]. This can be thought of a list of players that can exist on battlefield. On the diagram, attribute values of one object from each player type is provided. The values can be stored in this way. Also, a run-time value of an attribute can be provided to depict the situation of an entity just at a given time. Objects can be used in activity diagrams to specify momentarily state of an entity.

There are some other object elements used in CM diagrams. They are created to specify instances of some entities that are associated with missions and tasks. Examples of these are, terrain, tactical area, tactical line and tactical point objects used to specify places and properties of missions and tasks shown in ***Figure 28: MS MisSp Diagram 1***, ***Figure 29: MS MisSp Diagram 1.1***, ***Figure 30: MS WF Diagram 1.1*** and MS WF Diagram 1.2. If required, they can also be transformed and placed in UML object diagrams.

4.4.6.3 Component Diagrams

Component diagrams are used to show the software components, relations and interfaces between components. Components are high level abstractions, and any artifact interacting with main components can be depicted as components. Component diagram of SES project developed using SS EO, SS WF and SS ER diagrams is shown at ***Figure 42: HLD Component Diagram 1***.

First, software components of the system are determined by using SS EO diagram. SES_SW is the component developed in this project; so it is determined as the main component. Inside SES_SW, parts of the entity are placed as sub-components. HLA

and FCS are other high level components interacting with SES_SW component. Lower level components of FCS are not placed on diagram because the project does not deal with internal structure of it. Done with SS EO diagram, information on interfaces can be collected from SS WF diagrams. For each input/output entity between tasks that are executed by sub-components of system, an assembly relation is placed between those sub-components. Example is PLSYS_DB entity in ***Figure 38: SS WF Diagram 1***. As this is an output of a task executed by SES_DBEDIT component, it is a provided interface of it; and as input of task executed by SES_SCNEDIT, it is a required interface for it. So, an assembly connection is placed between these sub-components. SCN_DB is an input used also by an external component, FMS. To depict this, a port is defined for SES_SW that connects to internal SCN_DB provided interface with a delegate relation. Then, the provided interface of port and required interface of FMS are connected with dependency.

TERRAIN is an input for three of sub-components. As there is no component providing such an output, apparently there is a component providing interface for it. A port is created on SES_SW to depict that relation. All of the subcomponents of SES_SW gets user inputs “via keyboard” in SS WF diagrams. To reflect this, actor is placed as a component with provided “keyboard input” interface. All subcomponents are connected to keyboard input port of SES_SW with delegate relations. Lastly, ***Figure 37: SS ER Diagram 2*** is used in component diagram. This diagram depicts input/output relations with high level components of system. Using information on this diagram, two interfaces for SES_SW and FMS, and three interfaces for HLA are created. By means of these interfaces, it is observed that SES_SW and FMS do not communicate directly, but they communicate via HLA. Completing this part, component diagram of SES is finished, by using three types of CM documents.

4.4.6.4 Deployment Diagrams

Deployment diagrams show physical deployment of the system in hardware environment. Nodes are hardware elements on this diagram. Deployment diagram is developed by using SS EO, ER and WF diagrams. Deployment diagram of SES project is shown at ***Figure 43: HLD Deployment Diagram 1***.

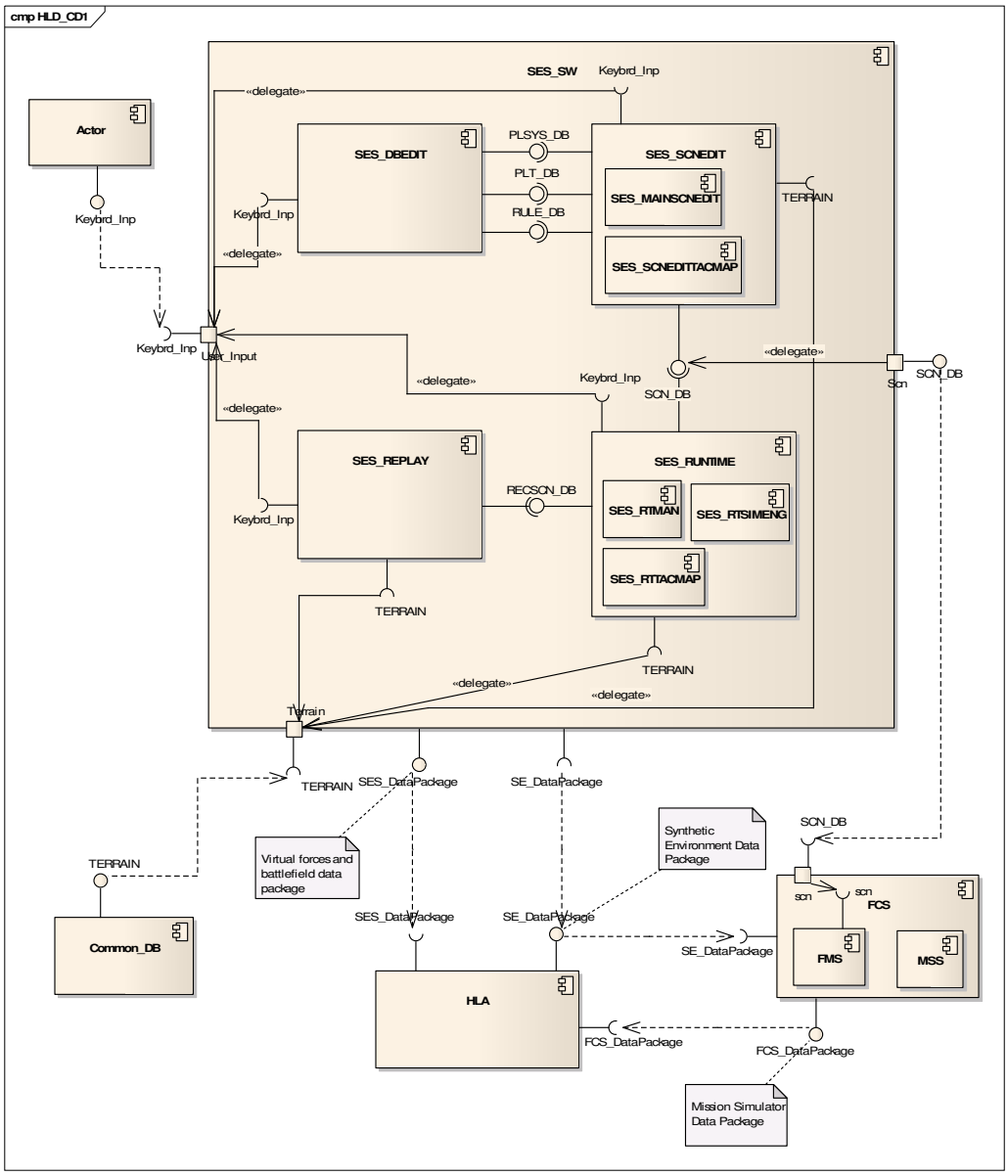


Figure 42: HLD Component Diagram 1

To develop this diagram, first, entities with kind facility are placed on deployment diagram as “nodes”. p/w relations among facility entities are used to determine hierarchy between nodes, and nodes are placed inside each other accordingly. Software components defined in component diagram are utilized. Existing components are placed inside nodes, to determine which components run on which hardware nodes; according to information provided by ***Figure 36: SS ER Diagram***

1. Input/output entities defined in SS WF and ER diagrams are placed as artifacts on deployment diagram. Artifacts are placed in the nodes together with components that give output to that artifact. Lastly, associations in ***Figure 37: SS ER Diagram 2*** is utilized to determine associations between SE_CC, NW_HW and FCC_HW. Also, SS WF diagrams are used to determine associations between nodes, according to components executing tasks and input/output relations. As a result, deployment diagram of system is developed. Looking at this diagram, one can grasp the physical structure of the system, and the relation of software components with hardware elements.

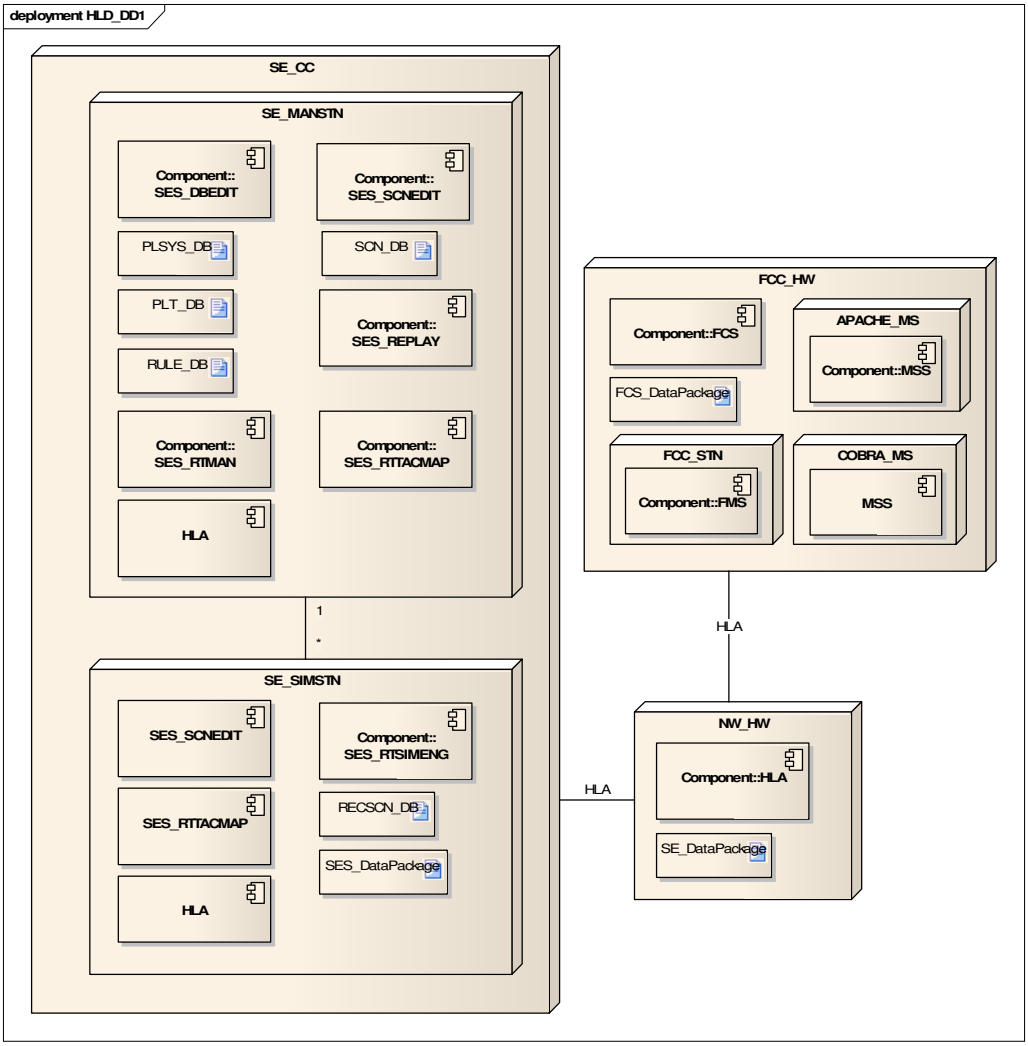


Figure 43: HLD Deployment Diagram 1

4.4.6.5 Use Case Diagrams

Use case diagram is a basic diagram in UML to capture the requirements of the system. Main functionality of the system and the interaction of the system with the users are identified in these diagrams. As use case diagrams depict interactions between external actors and functionalities, SS OS and WF diagrams are utilized to develop them. The developed use case diagram is depicted in ***Figure 44: HLD Use Case Diagram 1***. First, roles in SS OS diagram are placed as actors on use case diagram, like SE Manager, SE_SIMSTN Operator. Then, opening SS WF diagrams, each task realized by the actor (as specified in previous step) is added as a use case; and role realizing the task is associated with use relation on use case diagram. As an example, "generate DB records" task in SS WF diagram is placed as a use case. A use relation is generated between use case and SE Manager actor. While determining use cases, it is important to discard tasks that are not very meaningful in use case diagrams. For example, "start system", "start interface", "close system…" tasks are placed on SS WF diagrams to be able to explain how the activities are conducted on system in detail. It is not appropriate to place such tasks in use case diagrams, but rather use tasks that explain functionality of system. Finishing first level SS WF diagrams, tasks in lower level WF diagrams are transformed into use cases. Hierarchy among WF diagrams is reflected by means of include relations between use cases. It is important to be careful about details. For example, "SE Manager" uses the high level "Run and Manage Scenario" use case. But "SE_SIMSTN Operator" only uses a part of that use case, which is "Watch Scenario". Considering all these, UML use case diagram is completed. The dashed rectangles are placed to only increase the readability of diagram, they have no functional usage.

As discussed in ***3.8.5 Use Case Diagrams***, other than standard usage of use case diagrams as explained above, it is beneficial to use them for other aims in SDLC. In our case, use case diagram can be developed to show internal actors and use cases related to them, as shown in ***Figure 45: HLD Use Case Diagram 2***. For this aim, MS MisSP diagrams are utilized. First, all roles on mission space diagram are placed as actors on use case diagram. Then, missions connected to those actors with realize and responsible relations are placed as use cases, and they are associated with use

relations on use case diagram. Include and extend relations between missions are transformed into same relations between use cases. Lower level MisSp diagrams are also used to form use cases and include relations among them. Looking at the finalized use case diagram, one can understand basic internal actors of the system, and the functionalities they can conduct inside the system.

4.4.6.6 Activity Diagrams

Activity diagrams are used to display and provide details of many activities conducted in the system, which are provided in use case diagrams as high level functionalities. With a similar objective and notation, conceptual model has MS and SS work flow diagrams; and these diagrams are utilized to generate activity diagrams of high level design. As development of activity diagrams from CM work flow diagrams is straightforward, not all activity diagrams that can be developed from existing WF diagrams are depicted in this study. Some more examples of activity diagrams can be found at [67]. As shown in diagrams, task in WF diagrams is used as action in activity diagrams. Control flows are placed in the same way. Conditions of tasks and decision points are also placed on activity diagrams, with initial and final points. Synchronization point of CM is used in the same way, with the name fork and join. As seen from diagrams, all properties of WF diagrams are carried to activity diagrams except actors and input/output elements. They are discarded from activity diagrams, as they do not exist in UML notation.

The basic usage of activity diagrams in UML is, to detail use cases determined in use case diagrams. As discussed before, basically, use case diagrams include external actors of the system and use cases that are executed by those actors. For this reason, standard use case diagrams are developed for SES by using SS diagrams. Because of the same reason, it would be appropriate to develop standard activity diagrams from SS WF diagrams. But in practice, usage of activity diagrams is not restricted in this way. To get the most from CM, in this study, MS WF diagrams are also converted to UML activity diagrams and utilized to explain internal functionalities of the system. In this way, a set of UML activity diagrams is available in high level design of SES.

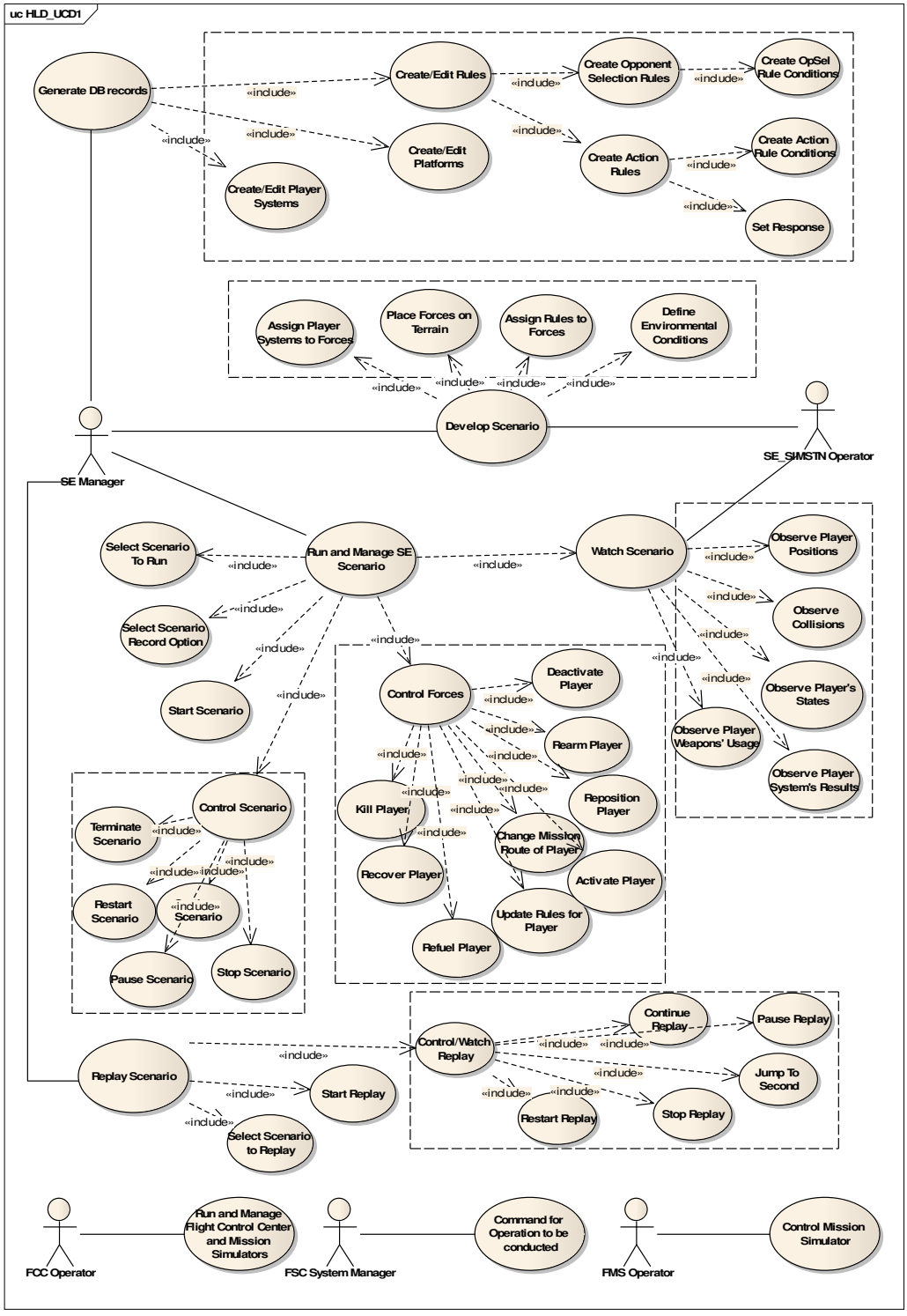


Figure 44: HLD Use Case Diagram 1

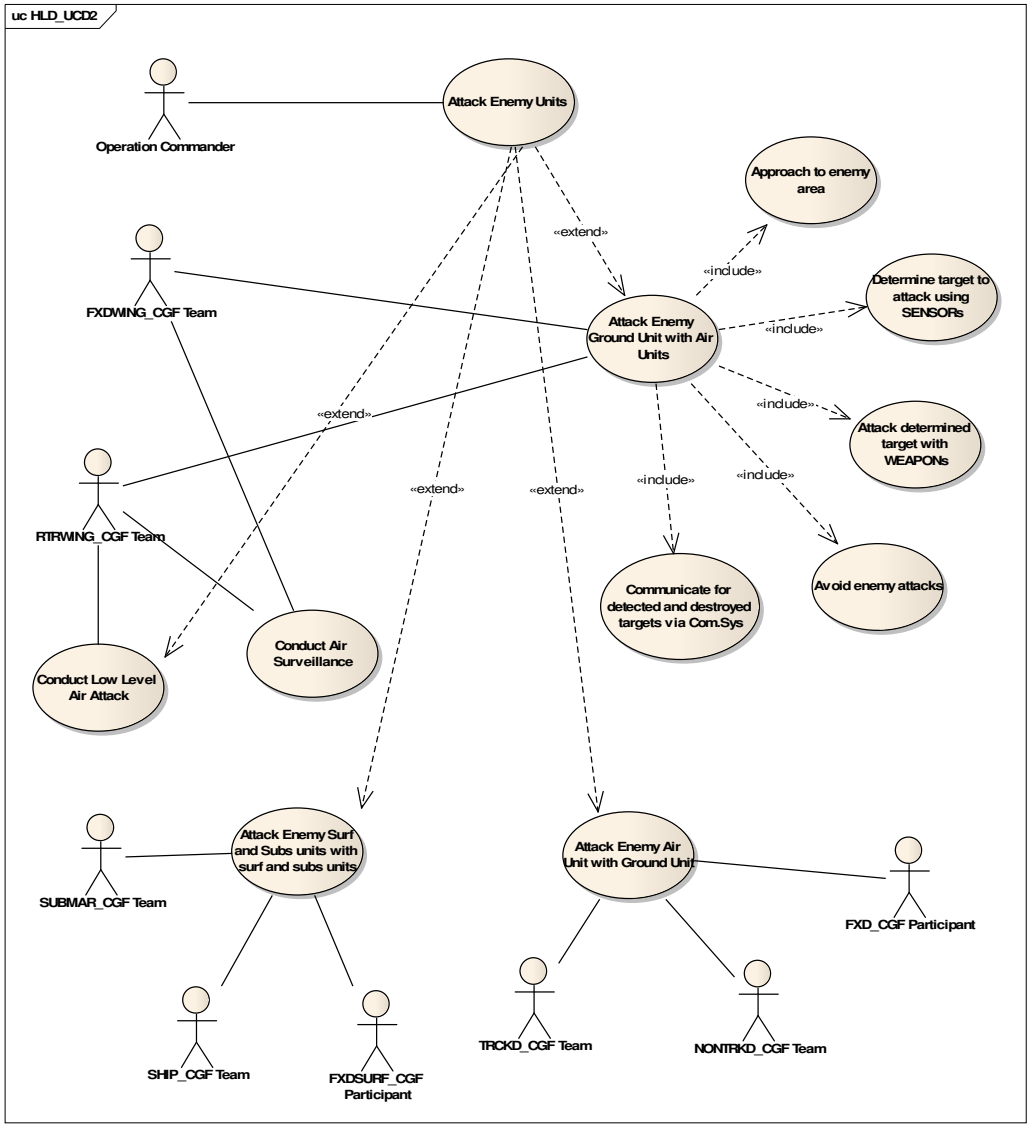


Figure 45: HLD Use Case Diagram 2

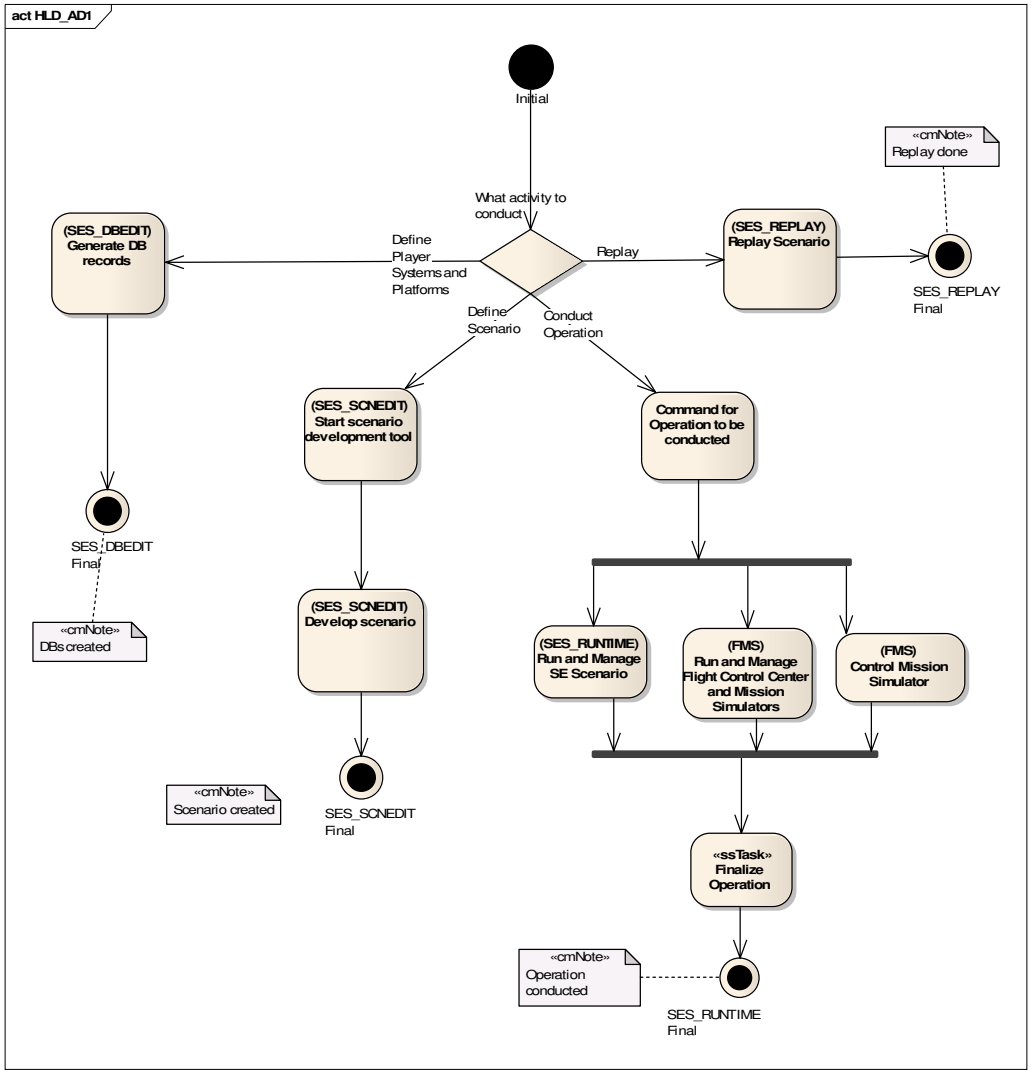


Figure 46: HLD Activity Diagram 1

4.4.6.7 State Machine Diagrams

State machine diagram models different states of a class in the system, and the transitions of that class between its states. CM entity state diagram describes the states of an entity and transitions between them. As entities are transformed into classes in UML, state machine diagrams and ES diagrams are very similar to each other. CM ES diagrams can be used directly, just after changing CM states to UML states and CM initial and final points and relations to UML points and relations.

Trigger events can also be used as they are. An example state machine diagram adapted from CM ES can be found at [67]. It is very similar to MS ES diagram developed in ***4.4.3.4***.

4.5 Summary and Discussion of Findings for Case Study

In the previous sections of this chapter, a comprehensive study is conducted to develop conceptual model of SES project, and to develop high level design of the project by using CM. The resulting CM diagrams and UML design diagrams are reported, and explanations about how they are developed and what they describe are provided in relevant sections. Being completed the implementation part of the case study on SES, in this section, findings of the implementation will be discussed by answering the research questions asked before starting implementation in section ***4.2.1 Research Questions***.

The first question was on utilizing and extending KAMA methodology to develop conceptual models of simulation systems other than C4ISR. To be able to answer this question, suggested activity was developing mission space CM for a synthetic environment project, by using proposed methodology as explained in ***CHAPTER 3***. Outcome of this activity, mission space CM of SES, is documented in section ***4.4.3***. Because of variety of player and player system types in SES project, a high number of entity ontology diagrams are developed. The important point in EO diagrams is that, high level entities are first defined as "information". In a hierarchical manner, entities are detailed from generic platforms to specific types like air platforms and lastly to most specific platforms like fixed wing air platforms. After detailing down to most specific player type, entities as forces (CGF) are created, that will be placed on battlefield. At every level of entity definition, common attributes relevant for that level are provided. Simulation space part of entities is also added by defining attributes that belong to SS. It is only the last level of entities, forces, that behaviors are added in entity definition. This is not a must, but rather a result of modeling approach followed, as only entities in the last level are "real forces", so have functions on their own. By adding parts of entities (like tail, body) and defining any

other entities using forces (like formation) the description of entities is completed. However, it is beneficial to also add specific force types in the name of objects. With the names of specific players that the user wants to use in battlefield, objects, and providing specific attribute values for them where available, EO diagrams for players are completed in lowest detail level.

Player systems are defined as a "part" of platforms at the highest level. Similar approach is followed also by player systems, as lower level player systems are defined as "equipment or material", and behaviors are carried by these entities. Especially detailed structure of sensors is reflected by defining modes as a part of sensor, and introducing different possible modes. By means of behaviors of sensors, it is possible to understand which sensor can detect which target types. Some weapon's complex structure, like missile and torpedo, is also defined by means of seekers defined as part of the entity. Sensors like sonar and radar that are carried by weapons are explained by giving cross reference from EO diagram describing them.

Rule is also a part of a platform. It is a specific entity for synthetic environment systems, because those systems do not have "real" players, but "virtual" players that are created by computer. These forces decide on how to behave on battlefield on their own, considering changing conditions. They do this by means of rules. This is why rules are very important in SE systems. There are two kinds of rule sets. By means of first rule set, an opponent is selected; and with second set, all possible detailed actions of player under different conditions are identified. These actions may be maneuvering, formation actions and usage of every kind of player system. All these are reflected in CM by defining them as entities in EO diagrams.

Another important entity to be modeled in simulations is environment. There are many approaches to specifically create CM of environment. In this study, environment is modeled using EO diagrams. Environmental characteristics are defined as separate entities with specific attributes; and relations are created among them. In this way, it is easy to grasp what environmental factors are active in the system. This depiction of environment turns out to be practical in also other diagrams where it is required to define effects of environment on other entities and activities.

A complex command hierarchy diagram is not required by SES, as there are not real "soldiers" or "commanders" in the system. But this diagram type is utilized to define actors from which roles that execute simulation system functions will be derived; and to define different types of teams. In this way, although not including large information, this diagram type turned out to carry important knowledge.

Because of special situation about players in synthetic environment systems, as described in previous paragraph, organization structure diagram is utilized in a different way. Rather than defining roles associated with actors, the player types that compose teams defined in CH diagrams are defined. Although deviated from its original use, OS diagrams are used to provide necessary information in CM.

Although in C4ISR systems, ES diagrams were mostly used to define states of players, they were very useful to define states, state transitions and events triggering those transitions for sensors in SE system. Modes of sensors were defined in EO diagrams; but relation between states of sensors that occurs by activating different modes could not be explained there. By means of provided diagrams, the basics of how player systems work were clearly explained. Although used only for sensors in this implementation, it is clear that ES diagrams can serve useful environment to model other aspects in simulation systems, according to nature of the system.

Entity relationship diagram serves as a flexible diagram to model various relations of entities. In this implementation, EO diagrams are used to depict all entities and all "inherit" and "part/whole" relations. In ER diagrams, only new entities of different types, like algorithm, input/output; and other relations between them (except inherit and part/whole) are defined. In SES, ER diagrams mainly served to describe algorithms in the system, to associate algorithms to entities that use them, and to define environment inputs that the algorithm considers. In this way, important aspects of synthetic environment are modeled in ER diagrams; and by utilizing flexible structure of these diagrams, many other issues may be modeled as required in different simulation systems.

Mission space diagram is very important in CM. As the objective of SES is to conduct different operations in battlefield, mission space diagram is defined accordingly. Different types of operations that are conducted by different player types are specified as missions of the system. High level objectives and measures of those missions are specified. Each mission, as it defines an operation, is associated with a terrain. In details of a mission, lower level missions conducted by forces are described. In this way, it is aimed to cover different types of high level activities that can be conducted by forces in battlefield. Many versions of mission space diagrams can be created, by making small changes in conditions of operation; but developed diagrams aim to provide an idea of how generic operations can be conducted on battlefield, by using which more detailed and diverse operations can be created.

Lastly, work flow diagrams are developed to detail high level missions defined in mission space diagram. Other than providing details of how activities are conducted in detail in system, by an overall evaluation of activity diagrams, many other properties of the system can be understood. For example, as actors are associated with tasks with realize relations, capabilities of those actors can be grasped. By using existing WF diagrams, one can understand that rotary wing players can use weapon types of missile, bomb and gun; but fixed wing players can only use missile and bomb. Roles of other entity types in operation are also explained, for example, by using tactical lines and points as inputs to tasks. So, it is seen that activity diagrams are an essential part of CM for every type of simulation system, to explain operations of system in detail and to provide many properties of system.

To give an overall answer to first research question, it is observed that proposed methodology was very successful to provide a comprehensive mission space conceptual definition of SES project. By means of wide coverage of SES project in simulation domain, proposed methodology is tested in many ways. Indeed, including different perspectives and providing a language to describe different properties of a system; the method can be utilized successfully not only for synthetic environment systems but also for other simulation systems.

The second question asks the utilization of KAMA methodology to develop simulation space CM. To answer the question, it is stated that SS CM will be developed for SES project by using proposed methodology. Proposed methodology involves development of five diagram types in SS CM. Although these diagrams also exist in MS and include same elements, the usage of them for SS differs. First, EO diagram is developed. It is used to define high level "facility" and "software" components of the system. They are associated with p/w relations to each other. In this way, this diagram provides a high level view of physical structure of simulation system, which is essential to understand the system.

"Real" actors in system are defined in MS CH diagrams; SS OS diagrams use those actor definitions to create roles that execute operations on simulation system. Those roles are external actors of simulation system that interact with system. It is necessary to define these roles, as they will be utilized in other diagram types to define activities conducted in the system.

A simple entity state diagram is developed; that describes main states that simulation system can be in. These are two main states, offline and online, that are observed in many simulation systems. This diagram type can be utilized if there are other states in the system, for example states specific to components.

Being specified main components of the system, including hardware and software, it is important to also specify relations other than p/w between those entities. As in MS diagrams, EO diagrams of SS are used to define all elements of the system and to depict p/w and inherit relations between them. Then, by means of ER diagrams, rest of the relations between entities is described. These may include predefined relations like "used by, input, output", or generic relations named by developer according to needs of the system. One of the ER diagrams is used to show which software component runs on which hardware component; and which software components are running in which states of the system. This is important information, and making it not clear at early stages of development always creates problems between users and developers. Other ER diagram is utilized to specify input/output relations between

high level components of the system. It is clear that ER diagram is a very important part of CM, to define simulation specific properties of system.

Lastly, work flow diagrams for SES are defined. WF diagrams of SS describe how simulation system is run and how different activities can be conducted by using components of the system. By placing roles defined in OS diagrams and connecting them with tasks by realize relations, the roles of actors while conducting simulation activities are also specified. SS work flow diagrams are very important to explain how simulation system activities are conducted; the roles of actors in system; and inputs and outputs generated by the system and relations of them with tasks. The importance of WF diagrams becomes clearer when every information provided in those diagrams are utilized to develop design.

As a conclusion, by using proposed methodology to develop SS CM, a comprehensive CM is developed that explains many aspects specific to simulation. Examining developed SS CM diagrams as a whole, one can grasp a full picture of simulation system components, relations between them, and how the system works. Although the same diagram types explained in MS diagrams are used, the usage of those diagrams specified for SS turned out to be very beneficial to describe the simulation system, and it can be used effectively in other simulation system types.

The third question is, how KAMA methodology and notation can be utilized to develop high level design. To answer this, a high level design of SES project is developed and documented using the guidelines provided in ***CHAPTER 3***. Proposed methodology claims that it is possible to develop a set of basic UML design diagrams by only utilizing CM knowledge. According to the guideline, seven UML design diagram types are developed for SES using SES CM. Basic class diagrams of system are formed by using entity definitions of MS CM. By utilizing MS EO, MS ER and SS EO diagrams, not only class diagrams are developed, but also they are organized as packages. Although class diagrams basically utilize mission space information, if SS conceptual model was not developed, only the attributes and behaviors for MS would exist, which would require extra effort for developers to specify SS attributes and behaviors. Moreover, the specified attributes and behaviors

would not be as well-built as done in conceptual modeling, because a conceptual analysis will be tried to be conducted during design analysis. Many issues in CM is utilized to develop class diagrams, as a result, detailed class diagrams of the system are achieved; on which developers can directly start detailed design activities.

Object diagram of the system is developed by using object element definitions in MS EO diagrams. This type of diagram is important as it carries knowledge of possible specific player types in the system, and their attribute values. By considering objects in this diagram, developers can start early data collection activities.

A detailed component diagram of the system is developed by using SS EO, WF and ER diagrams. CM information is utilized in this diagram, and as a result, a comprehensive component diagram including all software components, inner and outer interfaces, and the relations between them is developed. Similarly, deployment diagram of the system is developed by using SS EO, ER and WF diagrams. Deployment diagram of the system is achieved, utilizing information of the same diagrams from a different perspective, and demonstrating relations between hardware and software components of the system.

Use case diagrams are very important for the system to define interactions of external actors with the system and reflect main functionalities of the system. For this aim, SS WF diagrams are exploited, and use case diagrams showing all external actors of the system and the use cases that can be implemented by those actors are defined. It is observed that, a complete and beneficial use case diagram is formed for SES. Moreover, to get more from CM, use case diagram for internal actors of the system is also developed. By means of this diagram, basic activities that can be conducted by internal actors of the system can be understood. This carries the aim of use case diagrams to reflect user requirements one step further.

Activity diagrams of SES are developed by using MS and SS WF diagrams. In fact, UML activity diagram is a version of CM work flow diagram with less element types. As a result, activity diagrams that detail use case diagrams are developed by just discarding actors and inputs and outputs from CM diagrams. Similarly, entity

state diagrams of CM and state machine diagrams of UML are very similar. Because of this, CM ES diagrams are used directly to compose UML state machine diagrams.

In the overall, considering HLD development guidelines and design diagrams of SES, it is observed that every piece of CM knowledge is highly exhausted to form UML design diagrams. As a result, successful design diagrams are achieved. At the rest of the design studies, developers can directly base their studies on these diagrams to develop detailed design. They need not change the structure of these high level diagrams, they can develop detailed design by just adding details to class, object, activity and state diagrams. Moreover, to develop rest of the diagram types, although CM is not directly used, knowledge obtained by CM will be very beneficial.

This study is not only important to show that "CM is input to design", as stated as a generic sentence in many studies; but also to be a solid evidence that high level design can be developed by using CM as a direct input.

The fourth and the last question is a general question to evaluate how CM development activity affects requirements analysis, design and development activities. This question will tried to be answered by using experiences of the author of this study during development of SES project and during development of CM and high level design for SES in this study; and the discussions conducted with SES project development personnel. Handling requirements analysis activity first, SES project had a very long and problematic requirements analysis phase. There were many problems in user requirements obtained by user as the technical contract; and to clarify them, much effort was spent by the developers. However, trying to develop a set of written requirements, many issues could not been clarified or completed, and lead to continuous discussions between users and developers. All the developers of SES agree that, if a CM of the system as proposed in this study was developed parallel to requirements analysis activities, it would be much easier to meet at a conclusion about the properties of the system and requirements. When users see an easily understood model of the system, it is much easier for them to understand what developer means and make decisions on the point. As a result of discussions, it is obvious that conceptual model would be very helpful in determining and fixing

requirements, it would increase the quality of requirements, and it would not extend already prolonged requirements analysis phase.

The total time of developing SES CM diagrams by the author of this study was about one and a half month. Considering that the author used finalized requirements; if CM was developed inside SDLC, project team would not have the current domain knowledge; and meetings and discussions with the user would be conducted during CM development; it can be estimated that it would take the technical development team of three personnel up to two months to develop CM. Although requirements analysis and design phases of the project were mixed to each other, design phase of the project took six to seven months of the project team. The proposed CM development methodology and the resulting CM of SES project is evaluated with SES project development team. The evaluations are conducted with development team, which are software engineers that took place in requirements analysis, design and development phases of the project. These personnel do not have managerial level responsibilities in the project. The discussions are conducted with three engineers, which have three years of experience in average in military simulation field. This development team agrees that, if a high level design was developed using CM, and rest of design activities were based on CM knowledge, design phase could be shortened one to two months; because some basic design decisions about the system would have already be given, data would have been collected and basic system level judgments would be agreed on with the user. The most important of all, as the quality of the requirements would have been increased and an agreed CM would have been used; there would be fewer errors in design, and there would be less need to go back and change requirements, and update design accordingly after that. Of course, this would affect also implementation phase. With a design with fewer errors, implementation would be healthier, and later updates because of errors in design and unresolutioned requirements would be less. As a result, it seems that, considering the time, CM pays for the time spent during development of itself at later stages, at about the same amount. But the quality of requirements and design, and accordingly implementation of the project is expected to be higher, fewer errors are expected to be encountered and a simulation system that better meets user requirements is expected to be developed. Moreover, as the time passes and a common repository of

CM is developed, project developers will be able to reuse existing conceptual models from previous relevant projects. In this situation, CM can also save considerable time in development cycle. As a result, considering experiences of SES development team and discussions on developed CM, it can be concluded that a good implementation of CM in SDLC will considerably increase the efficiency of requirements analysis and design, by both decreasing total time of development and increasing quality. To be able to reach more comprehensive results about effects of CM development on SES development, other options were to conduct evaluations with and provide questionnaires for different stakeholders of project, like managers, quality engineers, main integrators and users of the project. Such a study was not conducted because of constraints of this study and physical difficulty of reaching all related stakeholders.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter provides a conclusion of studies provided in previous chapters, and suggests possible future work areas in conceptual modeling of simulation systems concluded as a result of this study.

5.1 Conclusion

This study includes a proposed methodology based on KAMA, which already suggests a complete methodology to develop and document mission space conceptual models, which basically aims to utilize conceptual model to develop requirements. This thesis study proposes an extended methodology for KAMA, to use the methodology to develop simulation space conceptual model and utilize developed mission space and simulation space conceptual model to develop high level design of the system.

As a result of literature review conducted in the study, it is concluded that KAMA is an outstanding approach as it provides a complete methodology to develop mission space conceptual models, including diagramming techniques that cover different perspectives of a system. Another conclusion is that, although most of the studies state that simulation space conceptual model is an essential part of conceptual model, and they emphasize the importance of conceptual model to be used in design; none of the studies provides a method to develop simulation space conceptual model, or

gives any solid explanation on how to utilize conceptual model in design phase of simulation development. There seems to be a gap in the field in these points, which determines the scope of this study.

To evaluate the proposed extended methodology, a case study is conducted on a synthetic environment project. The research questions and answers provide a comprehensive analysis of proposed methodology and case study implementation. The first research question aims to answer how extended KAMA methodology suits development of conceptual model to a system other than C4ISR. Considering mission space conceptual model diagrams developed using the proposed methodology, it is observed that an extensive model is formed for SES. Large number of entity ontology diagrams is developed, that reflect comprehensiveness of the system. Command hierarchy and organization structure diagrams are adapted easily to show relevant information for synthetic environment. Entity state diagrams are utilized to depict how different systems work. Entity relationship diagrams are used to depict different relation types between entities. Mission space diagrams reflect high level objectives of the system, including related relations with entities like actors, measures. Work flow diagrams are used widely to describe different operations that can be conducted in the system. Considering all diagrams, the author of this study thinks that many "hard-to-explain" characteristics of the system are described effectively. Examples are platform definitions and forces created from them, environment modeling, how players and player systems are affected from environment, basic functionalities provided by system (like collision, dynamics) and how entities utilize them, how player systems work, how players utilize player systems and rules of players. Although all of these concepts are hard to identify and understand in the system, they become clearer by means of conceptual model. This shows that proposed KAMA methodology is effectively utilized to develop mission space conceptual models for a simulation system other than C4ISR.

Second research question deals with how to develop simulation space conceptual model. Simulation space model is developed for SES using the proposed methodology. Although simulation space conceptual modeling is a new approach, the resulting diagrams have provided a clear explanation of simulation system. Entity

ontology diagram identifies facility and software entities of the system, and relations between them. Organization structure diagram is helpful to define roles related with simulation system. Entity state diagram is used to define states of main system. Entity relationship diagram is utilized to depict how software entities run on facilities. By means of work flow diagrams, all tasks conducted in the system, related roles, related entities, inputs/outputs and user interactions are defined. Considering all diagrams, basic simulation system properties are identified from different perspectives, which can be understood by both user and developer.

The third research question carries the aim of evaluating usage of extended KAMA methodology to develop high level design. Seven types of UML design diagrams are developed, by utilizing conceptual model information directly. The resultant diagrams are mature enough to continue detailed design activities over them; and all information in conceptual model is utilized in design. Developing high level design by utilizing conceptual model provides the project a solid design in harmony with requirements of the user. The developers can be sure about correctness of high level decisions inside design; and they need not spend any effort to give such decisions at design level. The well-formed design diagrams assure that proposed methodology proved to be helpful to utilize conceptual model to develop design.

Each extension identified in the study proved to be helpful somewhere in case study. As a specific example, a new element, algorithm, is utilized to define functionalities provided by system and implemented by different players. The element is helpful to identify behaviors in system and relations between attributes. Later, algorithms helped to define interfaces in design. Similarly, other extensions were beneficial both to provide conceptual description of some properties in the system and to develop high level design.

Evaluation of author's experiences and discussions with project development personnel reveals that development of conceptual model using extended KAMA methodology, and using it to develop high level design increase quality of end products, decrease errors in requirements analysis, design and development activities in SDLC; and can even decrease total development time if models are reused.

As a result, this study has made contributions to the field of simulation conceptual modeling by providing methodologies to develop simulation space conceptual model and to use conceptual model in simulation design activities. By means of the proposed extensions in this study, KAMA goes one step further to provide a complete methodology on developing conceptual model and utilizing it in simulation development life cycle. Considering the lack of applications in conceptual modeling field, the developed conceptual model is also an important application example in the field. The case study research is a complementary part of the study, as it assured that the proposed methodology could be used effectively in practice, and conceptual model served as an effective tool to increase the quality of activities and to decrease the errors in simulation development life cycle.

5.2 Future Work

Although the proposed methodology aims to provide a complete description of conceptual model development and documentation activities, there are possible future works that can be conducted to mature the study. In this study, the proposed methodology is evaluated by means of single case study, considering the comprehensiveness of the project to test the method and volume limitations of the study. However, to assure the completeness of the method, new case studies on different simulation systems can be conducted. By means of new studies, the proposed method's appropriateness for other simulation systems can be evaluated, and the proposed method can be enhanced and even be more standardized.

There are two important relations mentioned in this study; the transition from mission space conceptual models to simulation space conceptual models, and transition from conceptual model to high level design. Theoretical aspects of existence and uniqueness of these relations must be studied in more detail. Also, verification of simulation space conceptual model as well as mission space conceptual model and high level design require more theoretical study. Additional case studies from different military domains and more theoretical study on these

aspects considering different approaches in literature shall be conducted to assure uniqueness of the aspects and verify them.

To conclude, it is necessary to develop other case study applications of the methodology, to make more comprehensive evaluations and to fix the methodology as a standard.

Within the scope of KAMA study, a conceptual modeling tool is developed. This tool is not used in this study, because it was not completed yet. The development plan of that tool includes the aspects of original KAMA methodology. To extend KAMA as explained in this study, firstly, the extensions defined for developing mission space conceptual models shall be included in the tool. Secondly, a new part in the tool shall be included specifically to develop simulation space conceptual models, as suggested in this study.

In this study, a guide is proposed to develop high level design by using conceptual model. Although it is a guideline, the knowledge of conceptual model is transitioned as straightforward as possible, and the decisions of the developers are included in the least amount to compose design diagrams. A study can be conducted to automate the process of developing UML design diagrams. Even a supplementary tool can be developed for KAMA tool to automatically process completed conceptual model and compose design diagrams; and to provide an interface for developer to make corrections on the extracted high level design diagrams, before starting detailed design activities.

By means of such future work, it can be possible to totally automate the simulation development life cycle by means of models, starting from conceptual analysis and continuing with design and development. This has been the dream of many researchers in the area even in times that no standards existed in software development. The author of this study hopes that her study serves as a small step in the simulation field to realize the ultimate goal of automating the whole development process.

REFERENCES

[1] Meriam-Webster Online Dictionary, "Definition of Simulation," October 2007, http://www.m-w.com/dictionary/simulation.

[2] R. M. Chapman, "Conceptual Modeling Framework for Complex Synthetic Systems – an Example from F-15C Distributed Mission Training," in *Proceedings of Spring Simulation Interoperability Workshop*, 2000.

[3] J. J. Borah, "Conceptual Modeling- The Missing Link of Simulation Development," in *Proceedings of Spring Simulation Interoperability Workshop*, 2002.

[4] D. K. Pace, "Conceptual Model Development for C4ISR Simulations," in *Proceedings of the 5th International Command and Control Research and Technology Symposium*, October 24-26, 2000.

[5] S. Robinson, "Conceptual Modeling For Simulation: Issues And Research Requirements," in *Proceedings of the Winter Simulation Conference*, 2006.

[6] D. K. Pace, "Ideas About Simulation Conceptual Model Development," *Johns Hopkins APL Technical Digest*, vol. 21, Number 3, pp. 327-336, 2000.

[7] Defense Modeling and Simulation Office (DMSO), *Conceptual Models of the Mission Space (CMMS) Technical Framework USD/A&T-DMSO-CMMS-0002 Revision 0.2.1*, 13 Feb 1997.

[8] VV&A Product Development Group of the SISO Standards Activity Committee, *Draft Recommended Practice for Verification, Validation, and Accreditation of a Federation, an Overlay to the High Level Architecture Federation Development and Execution Process*, October 2006.

[9] L. Yilmaz and T. I. Ören, "A Conceptual Model for Reusable Simulations Within a Model-Simulator-Context Framework," in *Proceedings of CMS 2004 - Conference on Conceptual Modeling and Simulation,* October 28-31, 2004.

[10] H. Xue, Z. Lei and H. Ke-di, "Improving Simulation Conceptual Model," in *Proceedings of Fall Simulation Interoperability Workshop*, 2004.

[11] Defense Modeling and Simulation Office (DMSO), *Key Concepts of VV&A, RPG Special Topic*, 15 September 2006.

[12] D. K. Pace, "Development and Documentation of a Simulation Conceptual Model," in *Proceedings of Fall Simulation Interoperability Workshop,* 1999.

[13] R.M. Chapman, "Development and Documentation of a Simulation Conceptual Model," in *Proceedings of Spring Simulation Interoperability Workshop*, 2000.

[14] J. S. Weiner and P. D. Gutierrez, "Conceptual Modeling of a Legacy Constructive Simulation: A Use Case," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[15] S.Y. Harmon and S. M. Youngblood, "A Proposed Model for Simulation Validation Process Maturity," in *Proceedings of Spring Simulation Interoperability Workshop*, 2003.

[16] U. Eryılmaz, S. Bilgen and O. Molyer, "Verification And Validation Methods For Conceptual Modeling Of Mission Space," in *Proceedings of Savunma Teknolojileri Kongresi Ankara*, 2006.

[17] K. Hunt, "Exploiting the Relationships between CMMS Object Models and HLA Object Models to Facilitate Federation Design," in *Proceedings of Spring Simulation Interoperability Workshop*, 1998.

[18] D. H. Timian, R. T. Dunbar and H. P. Stickley, "Army Objective Force Intelligence, Surveillance, and Reconnaissance (ISR) Architecture Modeling Shortfalls," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[19] A. Tolk and J. A. Muguira, "The Levels of Conceptual Interoperability Model," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[20] YEROOS University of Louvain, "Yet another Project on Evaluation and Research on Object-Oriented Strategies," October 2007, http://yeroos.isys.ucl.ac.be/yeroos.html.

[21] C. Firat, "Conceptual Modeling And Conceptual Analysis In HLA," in *Proceedings of Fall Simulation Interoperability Workshop*, 2000.

[22] P. Gustavson, P. Zimmerman and C. Turrell, "Capturing Intent-Of-Use Meta-Data for the Conceptual Model –A Key to Component Reuse," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[23] Defense Modeling and Simulation Office (DMSO), *Conceptual Model Development and Validation RPG Special Topic*, 15 September 2006.

[24] L. W. Lacy, S. Youngblood and R. Might, "Developing a Consensus Perspective on Conceptual Models for Simulation Systems," in *Proceedings of Spring Simulation Interoperability Workshop*, 2001.

[25] D. K. Pace, "The Value of a Quality Simulation Conceptual Model," *Modeling & Simulaton*, vol..I, No. 1, pp. 9-10, January-March 2002.

[26] R. G. Sargent, "An Overview of Verification and Validation of Simulation," in *Proceedings of Winter Simulation Conference*, 1987.

[27] P. K. Davis, "Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations," RAND, Santa Monica, CA, R-4249-ACQ, 1992.

[28] U.S. Office Of The Under Secretary Of Defense (Acquisition And Technology) Washington DC, *Modeling and Simulation (M&S) Master Plan*, October 1995.

[29] Johnson T. H., "The Conceptual Model of the Mission Space and Data Engineering Toolset," Tech. Rep.

[30] J. Sheehan, T. Prosser, H. Conley, G. Stone, K. Yentz and J. Morrow, "Conceptual Models of the Mission Space (CMMS): Basic Concepts, Advanced Techniques, and Pragmatic Examples," in *Proceedings of Spring Simulation Interoperability Workshop*, 1998.

[31] M. Lundgren, M. G. Lozano and V. Mojtahed, "CMMS under the Magnifying Glass – An Approach to Deal with Substantive Interoperability," in *Proceedings of Fall Simulation Interoperability Workshop*, 2004.

[32] Defense Modeling and Simulation Office (DMSO), *High Level Architecture Federation Development and Execution Process (FEDEP) Model Version 1.5*, December 1999.

[33] H. S. Sarjoughian and B. P. Zeigler, "The Role of Collaborative DEVS Modeler in Federation Development," in *Proceedings of Fall Simulation Interoperability Workshop*, 1999.

[34] IEEE Computer Society, *1516.3™ IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)*, 23 April 2003.

[35] J. Graffagnini, S. Youngblood and R. Lewis, "An Overview of the Verification, Validation, and Accreditation (VV&A) Process for the HLA FEDEP," in *Proceedings of Spring Simulation Interoperability Workshop*, 1999.

[36] K. Ford "The Euclid RTP 11.13 Synthetic Environment Development and Exploitation Process (SEDEP)," *Virtual Reality Journal*, vol. 8, Number 3, pp. 168-176, June 2005.

[37] K. Ford, "The Euclid RTP 11.13 SE Development & Exploitation Process (SEDEP)," in *Proceedings of Spring Simulation Interoperability Workshop*, 2004.

[38] Euclid RTP 11.13, "SEDEP," October 2007, http://www.euclid1113.com/, SEDEP .

[39] A. Lemmers and M. Jokipii, "SEST: SE Specifications Tool-set," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[40] M. Keuning and A. Lemmers, "RTP 11.13 Gets To Grips with SE Specifications," in *Proceedings of European Simulation Interoperability Workshop*, 1997.

[41] M. Brassé, O. M. Mevassvik and T. Skoglund, "Federation Composition Process and Tool Support in EUCLID RTP 11.13," in *Proceedings of Fall Simulation Interoperability Workshop*, 2003.

[42] V. Kabilan and V. Mojtahed, "Introducing DCMF-O: Ontology Suite for Defence Conceptual Modeling," in *Proceedings of European Simulation Interoperability Workshop*, 2006.

[43] V. Mojtahed, M. G. Lozano, P. Svan, B. Andersson and V. Kabilan "FOI DCMF -Defence Conceptual Modeling Framework," FOI Sweeden, November 2005.

[44] K. Porter, "JCMMS: Domain Knowledge for Developing Military Models," in *Proceedings of Fall Simulation Interoperability Workshop*, 2000.

[45] S. Risner, K. Porter, L. Lacy, L. O'Brien and G. Kollmorgen, "Conceptual Modeling in the Joint Simulation System (JSIMS)," in *Proceedings of Fall Simulation Interoperability Workshop*, 1998.

[46] M. L. Metz, "Comparing the Joint Warfare System (JWARS) Conceptual Model to a Conceptual Model Standard," in *Proceedings of Fall Simulation Interoperability Workshop*, 2000.

[47] F. L. Dougherty, Weaver F. and Cluff M. L., "Joint Warfare System Conceptual Model of the Mission Space," in *Proceedings of Spring Simulation Interoperability Workshop*, 1998.

[48] A. J. Duck, D. H. Timian, M. R. Auth, R. T. Dunbar and C. R. Karr, "Army Future Force Intelligence, Surveillance, and Reconnaissance (ISR) System-of-System (SoS) Conceptual Modeling," in *Proceedings of Fall Simulation Interoperability Workshop*, 2004.

[49] J. J. Borah, "Conceptual Modeling- How Do We Do It?—A Practical Example," in *Proceedings of Spring Simulation Interoperability Workshop*, 2003.

[50] J. Atherton and M. Jones, "Conceptual Modelling & Dataset Repositories: Tools to support SE Process," in *Proceedings of Fall Simulation Interoperability Workshop*, 2006.

[51] S. Bilgen, O. Demirörs, K. İmre, V. İşler, A. Karagöz, O. Molyer and A. Erçetin, "Conceptual Modeling Of C4ISR Mission Space," in *Proceedings of USMOS*, 2005.

[52] N. A. Karagöz, O. Demirörs, Ç. Gencel and Ç. Ündeğer, "Mission Space Conceptual Model Development In The Simulation Systems: A Process Definition," in *Proceedings of ODTÜ Savunma Teknolojileri Kongresi Ankara*, 29-30 June 2006.

[53] N. A. Karagöz and O. Demirörs, "A Model Driver Approach to Conceptual Model Development for Simulation Systems," in *Proceedings of USMOS Ankara*, 2-3 June 2005.

[54] N. A. Karagöz and O. Demirörs, "A UML Based Metamodeling Framework for Mission Space Conceptual Model Development," in *Proceedings of Savunma Teknolojileri Kongresi Ankara*, 29-30 June 2006.

[55] D. Brade, "Conceptual Modeling Meets Formal Specification," in *Proceedings of Spring Simulation Interoperability Workshop*, 2003.

[56] B.P. Zeigler, *Theory of Modeling and Simulation.* New York: Wiley, 1976.

[57] A. Tolk, "Composable Mission Spaces and M&S Repositories - Applicability of Open Standards," in *Proceedings of Spring Simulation Interoperability Workshop*, 2004.

[58] IEEE Computer Society, *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications,* Los Alamitos, CA, October 1998.

[59] K. E. Wiegers, *Software Requirements.*, Microsoft Press, 2003.

[60] V. T. Dobey, "System Contextual Development of Physical Environmental Representations," in *Proceedings of Spring Simulation Interoperability Workshop*, 2004.

[61] Object Management Group, *Unified Modeling Language: Superstructure Version 2.1.1*, 03 February 2007.

[62] MISQ Discovery, "Quantitive Research in Information Systems," October 2007, http://www.qual.auckland.ac.nz/.

[63] Y.K. Yin, "Case Study Research: Design and Methods Third Edition," SAGE Publications, USA, 2002.

[64] I. Benbasat, D. K. Goldstein and M. Mead, "The Case Research Strategy in Studies of Information Systems," *MIS Quarterly*, vol. 11, No. 3. pp. 369-386, September 1987.

[65] R. Ayres and M. R. Moulding, "A Pilot Language for Conceptual Modeling of the Battlespace," in *Proceedings of Spring Simulation Interoperability Workshop*, 2001.

[66] D.W. Blake, C. Little and J. Morse, "The Navy's Probability of Raid Annihilation Assessment Process Standards & Architecture and Systems Engineering Concept Model," in *Proceedings of Fall Simulation Operability Workshop*, 2003.

[67] B.E. Aysolmaz, "Conceptual Model of a Synthetic Environment Simulation System Developed Using Extended KAMA Methodology," METU Informatics Institute, Turkey, Tech. Rep., METU/II-TR-2007-17, November 2007.