GOAL ORIENTED MODELING OF SITUATION AWARENESS IN A COMMAND AND CONTROL SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HASAN ALİ SOĞANCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

NOVEMBER 2010

Approval of the thesis:

GOAL ORIENTED MODELING OF SITUATION AWARENESS IN A COMMAND AND CONTROL SYSTEM

submitted by **HASAN ALİ SOĞANCI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Dr. Halit Oğuztüzün _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Faruk Polat _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Halit Oğuztüzün _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Ahmet Coşar _____
Computer Engineering Dept., METU

Asst. Prof. Dr. Pınar Şenkul _____
Computer Engineering Dept., METU

Dr. Bülent Mehmet Adak _____
Software Engineering Dept., ASELSAN Inc.

Date: 24.11.2010

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 work.

Name, Last name: Hasan Ali SOĞANCI

Signature :

ABSTRACT

GOAL ORIENTED MODELING OF SITUATION AWARENESS IN A COMMAND AND CONTROL SYSTEM

SOĞANCI, Hasan Ali
M.Sc., Department of Computer Engineering
Supervisor: Assoc. Prof. Halit OĞUZTÜZÜN

November 2010, 89 pages

This thesis presents a preliminary goal oriented modeling of situation awareness in a command and control system. Tropos, an agent oriented software development methodology, has been used for modeling. Use of Tropos allows us to represent, at the knowledge level, the Command and Control actors along with their goals and interdependencies. Through refinement we aim to derive an architectural design for the Situation Awareness component of an Air Defense Command and Control system. This work suggests that goal oriented methodologies can be successfully used in the modeling of the complex systems at the requirement analysis phase. By analyzing dependencies between Command and Control entities, it should be possible to improve the modularity of the Command and Control system architecture.

Keywords: Command and Control Systems, Situation Awareness, Requirement Engineering, Goal Model, Tropos

ÖZ

BİR KOMUTA KONTROL SİSTEMİNDEKİ DURUM FARKINDALIĞININ AMAÇ TABANLI OLARAK MODELLENMESİ

SOĞANCI, Hasan Ali
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doçent. Dr. Halit OĞUZTÜZÜN

Kasım 2010, 89 sayfa

Bu tez, bir komuta kontrol sistemindeki durum farkındalığının amaç tabanlı olarak modellemesini sunar. Modelleme için ajan tabanlı bir yazılım geliştirme yöntemi olan Tropos kullanılmıştır. Tropos metodolojisinin kullanımı, Komuta ve Kontrol aktörlerinin amaçlarını ve birbiriyle olan ilişkilerini bilgi seviyesinde tasvir etmemizi sağlar. Bunu göstermek için Hava Savunma Komuta ve Kontrol Sisteminin Durum Farkındalığı bileşeninin mimari tasarımını elde etmeyi amaçladık. Bu çalışma, amaç tabanlı yöntemlerin karmaşık sistemlerin modellenmesinin gereksinim analizi sırasında başarılı bir şekilde kullanılabileceğini önerir. Komuta ve Kontrol varlıklarının aralarında ilişkiler analiz edilerek Komuta ve Kontrol sistem mimarisinin modülerliği arttırılabilir.

Anahtar Kelimeler: Komuta ve Kontrol Sistemleri, Durum Farkındalığı, Gereksinim Mühendisliği, Amaç Modeli, Tropos

To My Family

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks to my supervisor, Assoc. Prof. Halit OĞUZTÜZÜN for his efforts and guidance throughout this thesis work.

I also would like to thank my colleagues at ASELSAN SST-YMM department for their support throughout this thesis work.

Finally, I would like to thank my family for their support and patience.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

AOSE	Agent Oriented Software Engineering
AUML	Agent-Based Unified Modeling Language
BDI	Belief Desire Intention
BfG	Battlefield Geometry
C2	Command and Control
CASE	Computer-Aided Software Engineering
CD	Compact Disc
DoD	Department of Defense (US)
DVD	Digital Video Disc
GIS	Geographic Information System
IFF	Identification Friend or Foe
KAOS	Keep All Objects Satisfied
MaSE	Multiagent Systems Engineering Methodology
NATO	North Atlantic Treaty Organisation
PDT	Prometheus Design Tool
RTO	Research And Technology Organisation (NATO)
SA	Situation Awareness
TAOM4E	Tool for Agent Oriented visual Modeling for the Eclipse platform
TDL	Tactical Data Links
UI	User Interface
UML	Unified Modeling Language
US	United States
USB	Universal Serial Bus

CHAPTER 1

INTRODUCTION

With the introduction of the first programming languages, software development has undergone several evolutionary changes. These evolutionary changes provided opportunities for building more complex software systems. With the introduction of complex software systems, it is realized that software is difficult to deliver on time, within the available budget and with the required quality factors. To cope with this software crisis many approaches to software development were proposed.

Many different attempts were made to cope with software crisis. Some attempts improved programming languages, while others provided some CASE tools. Maybe the most effective attempts were the introduction of software development methodologies. In the recent years, Software Product Lines and Agent Oriented Methodologies have gained a wide popularity in software engineering.

1.1 Software Intensive Systems

Today's software systems are much more complex than earlier systems. Building software-intensive systems requires using appropriate engineering methods. There are four stages of building complex software systems [7]: Analysis, Design, Development and Deployment. First three of these are classical stages. Analysis is basically defined as the identification of the problem and domain, and, then, describing the problem and separating into smaller parts. At the design stage, solution architecture of the problem is defined. Practically, it consists of giving a solution principle of the problem. Development is the process of implementing the defined architecture, in other words, creating the solution. For the software systems,

development is generally consisting of coding. At the deployment stage, solution is integrated into real world. For example, combining software applications together and running them.

Software-intensive systems are becoming ever more distributed, heterogeneous and decentralized. Their size, and the complexity of the interactions within them and with people, will continue to increase. There exist different kinds of complexity in the development of software-intensive systems. Software-intensive systems have to operate in large and non-deterministic environments. The complexity arises with increase in the complexity of knowledge, interaction and adaptation.

Capturing and validating requirements is a crucial area for developing high quality software-intensive systems. Classical requirement engineering methods and tools are not powerful enough capturing the requirements of software-intensive systems. There is a need for an analysis method that supports both capturing and validating requirements.

Analyzing requirements of software-intensive systems, especially at the early requirements phase, requires high level abstraction. In other words, these systems must be analyzed at the knowledge level.

1.2 Modeling Command and Control Systems

Command and Control (C2) Systems are software intensive systems. Requirement analysis is a crucial phase of developing C2 systems. First activity of requirement analysis of C2 systems requires domain knowledge acquisition. Domain knowledge can be carried out through the analysis of documents, stakeholder knowledge and production of scenarios. Following this, stakeholders and their desires, needs, and preferences must be identified. These steps are essential in capturing and validating requirements. Early requirement must be captured at an abstract level. Therefore, requirement analysis of C2 systems must be at the knowledge level.

Goal oriented requirement analysis can deal with software-intensive systems. It helps to identify the domain stakeholders and their dependencies. By analyzing these dependencies, relations of system functionalities are better understood. Therefore, goal oriented analysis provides capturing and validating requirements.

This study aims to presents a preliminary goal oriented modeling of a C2 system. Tropos, an agent oriented software development methodology, has been used for analysis. Use of Tropos allows us to represent, at the knowledge level, the Command and Control actors along with their goals and interdependencies for requirements modeling.

Command and Control are fractal like concepts. C2 occurs at many levels of an organization from the enterprise level to mission level. We preferred to model C2 at a level between enterprise level and mission level. In the military scope, C2 can be at strategic level, operational level and tactical level. We preferred to model a C2 system at the tactical level.

C2 conceptual model is sufficient to model C2 systems in general with basic operations, i.e. command, control, sensemaking, actions. But, when trying to model C2 in more details, a specific type of C2 system must be chosen, because each system has different features. Thus we have chosen air defense system.

Modeling the whole air defense system at architecture level would cause the model be unmanageably large that we could not afford to model towards architectural design phase. Therefore, we decided to keep the scope of this work as situation awareness (SA) in air defense system, because it is prevalent in all C2 systems.

Tactical Picture helps the operator (i.e. officer in charge) to establish SA. Tactical Picture plays role in the first level of SA; perception of elements in the environment. The second and third levels of SA, comprehension and projection, are operator related operations.

As a result, this work presents goal oriented model of SA in a C2 system, namely, air defense system, especially Tactical Picture component which provides input for the first level of SA activities. This work suggests that goal oriented methodologies can be successfully used in the modeling of the complex systems at the requirement analysis phase. This thesis applies Tropos methodology from early requirement analysis phase to architectural design phase. By analyzing dependencies between Command and Control entities, it should be possible to improve the modularity of the Command and Control system architecture.

1.3 Organization of the Thesis

This thesis work includes 5 chapters. Chapter 2 gives the necessary background on goal oriented modeling and situation awareness in a command and control systems. Chapter 3 defines the proposed model of SA, modeled with Tropos methodology. Chapter 4 discusses the model and methodology, and gives information about related works. Chapter 5 concludes the thesis and points to future work.

CHAPTER 2

BACKGROUND

This chapter gives the necessary background on goal oriented modeling and SA in C2 systems. First, agent oriented methodologies are introduced. After that, key concepts and development phases of Tropos methodology are briefly described. Then, some well-known agent oriented software development methodologies are summarized and compared with Tropos, and then reasons for choosing Tropos methodology are explained. Final section explains the general concepts and conceptual model of C2 with emphasis on the SA process.

2.1 Agent Oriented Software Development Methodologies

Agents are typically described as possessing human characteristics, for example, agents are normally considered to be autonomous, adaptable, social, knowledgeable and mobile [16]. Agent-oriented software engineering (AOSE) [1] methodologies have received a great deal of attention in recent years. They aimed to provide tools and techniques for analyzing and designing agent-based software systems. Different methodologies are proposed, such as Gaia [3] and Tropos [5]. The term "Goal Oriented" is sometimes used instead of "Agent Oriented".

Key concepts for goal oriented methodologies are agent/actor and goal.

> **"Actor**, which models an entity that has goals and intentions within the system or the organizational setting. An actor represents a physical, social or software agent as well as a role or position." [5]

Role is an abstract characterization of the behavior of a social actor, and a position represents a set of roles, typically played by one agent [5]. An agent can occupy a position, while a position is said to cover a role.

According to [5], goal is defined as follows:

> **"Goal**, which represents actors' strategic interests. We distinguish hard goals from soft goals, the second having no clear-cut definition and/or criteria for deciding whether they are satisfied or not." [5]

AOSE can fill the gap between high-level organizational needs and system development [5]. Goals are valuable in identifying, organizing and justifying system requirements [9], whereas the notion of agent provides a quite flexible mechanism to model the stakeholders. Because agents and goals are suitable for modeling stakeholders' needs, Goal Oriented Requirement Engineering is more powerful than other requirement engineering methodologies. Agent and goal relations enable system analyst to easily find and justify the requirements of a system.

One of the main advantages of the goal oriented methodologies is that they allow capturing answers of *why* question beside the answers of *how* or *what* questions.

2.2 Tropos Methodology

Tropos [6] is a software development methodology, based on the i* model of Eric Yu [10]. The i* model has been applied in various application areas [11, 12, 13], including requirements engineering. Actors, goals and actor dependencies are the primitive concepts in an i* model [10]. The aim of i* model is to develop a modeling process which involves humans, software and hardware systems. In the early requirement analysis of i* one can understand that not only how and what, but also why a software is developed. The i* requirement analysis includes both system's functional and non-functional requirements. Tropos allows us to

understand the system's operating environment, and to understand the relations between system and environment.

The Tropos methodology supports software development process, from application domain analysis down to the system implementation. There are five development phases of Tropos methodology: *Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation* [5]. Detailed design and Implementation phases are not in the scope of this thesis.

2.2.1 Key Concepts

Key concepts of the Tropos methodology are actors, goals, plan, resource, dependency and capability [5]. Actor, models an entity that has goals and intentions within the system. An actor represents "a physical, social or software agent as well as a role or position" [5].

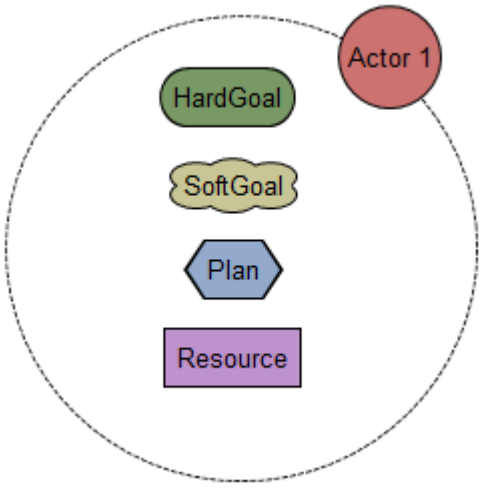


Figure 1. Tropos Modeling Elements.

Goal represents actors' strategic aims. Goals are divided into two: Hard goal and soft goal. Hard goals and soft goals are used to represent functional and non-functional requirements respectively. Soft goals do not have clear definitions. It is

not easily possible to decide whether a soft goal is satisfied or not. In the rest of the thesis, the term "goal" is used instead of hard goals; the term "soft goal" is used instead of soft goals.

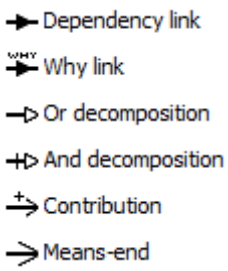


Figure 2. Links between elements in Tropos.

Plan is a way of doing something. Plan helps hard/soft goals to be satisfied.

Resource is "a physical or an informational entity" [5].

Dependency can be between two actors: "One actor depends on another actor through a plan, goal and resource of the latter. The former actor is called the *depender*, while the latter is called the *dependee*. The object around which the dependency centers is called *dependum*" [5].

Capability represents the ability of an actor to execute a plan for the fulfillment of a goal.

Figure 1, Figure 2 and Figure 3 depict Tropos modeling elements, links between elements, and dependency relations respectively.

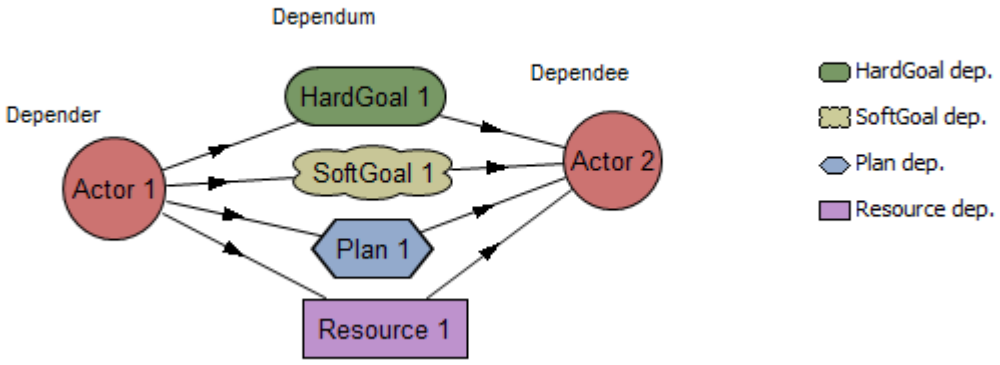


Figure 3. Dependency relations**.**

2.2.2 Modeling activities

Actor Modeling [5] consists of identifying actors. In each development phase, actor modeling focuses differently. In early requirement analysis phase, modeling focuses on application domain stakeholders and their intentions. During late requirement, actor modeling focuses on the definition of the system-to be. In architectural design, it focuses on sub-systems (actors), connected through data and control flows.

Dependency Modeling [5] consists of identifying dependency relations between actors. One actor depends on another actor by a goal, plan or resource. In early requirement analysis phase, modeling focuses on goal dependencies between social actors. During late requirement, modeling focuses on dependencies of the system-to-be actor.

A graphical representation of the model obtained following actor and dependency modeling activities is given through **actor diagrams**. Actor diagrams describe the actors, their goals and dependencies between actors. Example actor diagrams are presented in section 3.2.1 and 3.3.1.

Goal and Plan Modeling [5] focuses on identifying an actor's sub elements, such as goals, plans, resources and their relations. There are three basic relations within an actor: means-end analysis, contribution analysis, and AND/OR decomposition. Means-end analysis relates a plan/resource to a hard/soft goal. Contribution analysis identifies goals/plans that can contribute positively or negatively. Goals, plans and resources can contribute to hard/soft goals in the fulfillment of the goal. AND/OR decomposition combines AND and OR decompositions of a root goal into sub-goals.

A graphical representation of goal and plan modeling is given through **goal diagrams**. Example goal diagrams are presented in section 3.2.2 and 3.3.2.

Capability modeling [5] starts at the end of the architectural design. Before capability modeling system sub actors must have been specified in terms of their own goals and the dependencies with other actors. Each system's sub actor must provide capabilities for executing a plan for achieving its own goals. Actors should also provide capabilities for the dependencies with other actors. Goals and plans are part of the capabilities. In detailed design, each agent's capability is further specified and then coded during the implementation phase. A graphical representation of capability modeling is given through **capability diagrams**. A capability diagram is illustrated in Figure 4.a.

2.2.3 Development Phases

There are five development phases of Tropos methodology: *Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation* [5]. Detailed design and Implementation phases are not in the scope of this thesis.

2.2.3.1 Early Requirements Analysis

Early Requirements Analysis consists of identifying stakeholders and analyzing their intentions [17, 5]. Stakeholders are modeled as social actors who depend on one another for goals to be achieved, plans to be performed, and resources to be furnished. Intentions are modeled as goals. Once the stakeholders have been identified, model must be enriched with further details. Early requirement analysis of C2 system is presented in section 3.2.

2.2.3.2 Late Requirements Analysis

Late Requirement Analysis focuses on the system-to-be [5, 17]. In this phase, system-to-be is analyzed with its operating environment. The system-to-be is represented as an actor with dependencies with other actors. These dependencies define the system's functional and non-functional requirements. Late requirement analysis is discussed in section 3.3.

2.2.3.3 Architectural Design

Architectural Design phase focuses on the system architecture in three steps [5]. Firstly, overall architectural organization is defined. Sub-actors are introduced in the system. New actors may be added to model during goal analysis and after the choice of a specific architectural style. Dependency, goal and plan modeling activities are done for new actors. As a result of architectural design, extended actor diagram including new actors and their dependencies with the other actors are presented.

Second step consist of identifying capabilities needed by the actors to fulfill their goals and plans. Capabilities are identified by analyzing the extended actor diagram.

Finally, the last step consists of defining a set of agent types and assigning each of them one or more different capabilities.

Details of Architectural Design Analysis are discussed in Section 3.3.

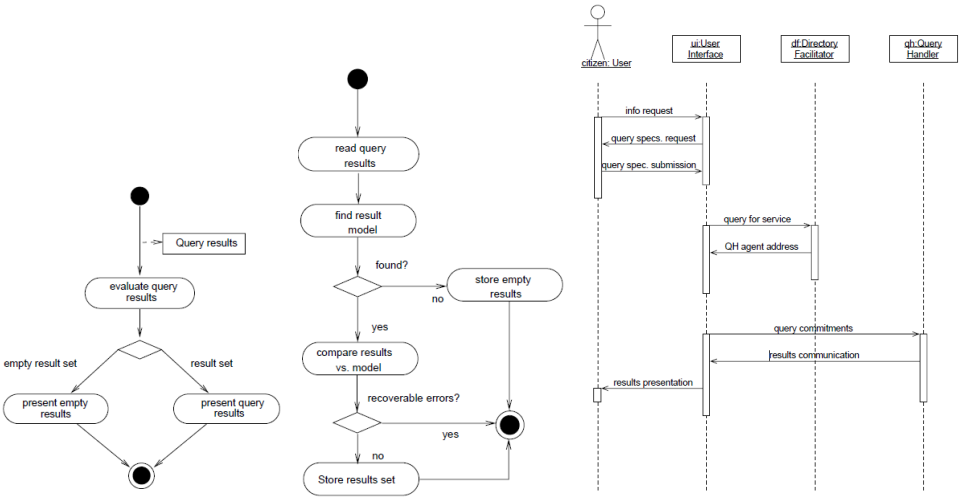
2.2.3.4 Detailed Design

The detailed design phase introduces additional details for each architectural component of a system [17]. This includes actor communication and actor behavior. Agents' goals and capabilities are specified in detail. At this phase, communication among agents is also studied in detail.

Practical approaches for this activity are usually proposed within specific development platforms In other words, this step is related to implementation choices.

Capability/plan models are represented with *capability diagrams* and *plan diagrams* respectively. UML activity diagrams can be used for capability and

plan diagrams. AUML diagrams proposed in [12] can be used for specifying agent interactions. Example diagrams are reproduced in Figure 4.



(a) (b) (c)

Figure 4. Example diagrams for Detailed Design phase. [5]

(a) Capability diagram represented as an UML activity diagram.

(b) Plan diagram represented as an UML activity diagram.

(c) Agent interaction diagram represented as an AUML sequence diagram.

2.2.3.5 Implementation

Implementation phase consists of coding of agents with a set of plans in order to make them capable of achieving goals.

> "The Implementation activity follows step by step, in a natural way, the detailed design specification on the basis of the established mapping between the implementation platform constructs and the detailed design notions." [5]

JACK development environment can be used at implementation phase. JACK [25] is an agent-oriented development environment designed to extend Java with the theoretical BDI agent model [26].

2.3 Some Other Agent Oriented Software Development Methodologies

There are several agent oriented software development methodologies proposed in the literature. In this section, we will overview some well-known methodologies, namely, Gaia, MaSE, Prometheus, KAOS and AUML. At the end of the section, we will state the reasons for why we have chosen Tropos.

2.3.1 The Gaia Methodology

Gaia [3] is a methodology for agent-oriented analysis and design. Gaia methodology proposes three phases: requirement, analysis and design. Gaia mostly focuses on analysis and design phases. Analysis and design in Gaia methodology can be thought of as a process of developing increasingly detailed *models* of the system to be constructed.

Gaia analysis phase models are the *roles model* and the *interaction model*. The *roles model* identifies the key roles in the system. Dependencies and relationships between the various roles in a multi-agent system are represented in the *interaction model*.

Gaia design phase models are the *agent model*, the *services model* and the *acquaintance model*. The purpose of the *agent model* is to document the various agent types that will be used in the system. The aim of the *services model* is to identify the services associated with each agent role, and to specify the main properties of these services. Service represents the function of the agent. The *acquaintance models* simply define the communication links that exist between agent types.

2.3.2 The MaSE Methodology

Multiagent Systems Engineering Methodology (MaSE) [22] is similar to Gaia with respect to generality and the application domain supported, but MaSE supports automatic code creation through the MaSE tool. The goal of MaSE is to lead the designer from the initial system specification to the implemented agent system.

The MaSE methodology is divided into seven phases. The first phase, *Capturing Goals*, constructs a structured hierarchy of system goals. *Applying Use Cases*, the second phase, creates use cases and sequence diagrams based on the initial system specification. The third phase, *refining roles*, creates roles that are responsible for the goals defined in the phase one. The fourth phase, *creating agent classes*, maps roles to agent classes in an agent class diagram. The fifth phase, *constructing conversations*, defines a coordination protocol in the form of state diagrams that define the conversation state for interacting agents. In the sixth phase, *assembling agent classes*, the internal functionality of agent classes are created. Selected functionality is based on five different types of agent architectures: Belief-Desire-Intention (BDI), reactive, planning, knowledge based and user-defined architecture. The final phase, *system design*, create actual agent instances based on the agent classes, the final result is presented in a deployment diagram.

2.3.3 The Prometheus Methodology

Prometheus [21] is a methodology for developing intelligent agents. The Prometheus methodology [21] includes three design phases.

> "The system specification phase focuses with inputs (percepts), outputs (actions) and any important shared data sources. The architectural design phase uses the outputs from the previous phase to determine which agents the system will contain and how they will interact. The detailed design phase looks at the internals of each agent and how it will accomplish its tasks within the overall system." [21]

Prometheus methodology is supported by design tool PDT. Tool allows a user to enter and edit a design, in terms of Prometheus concepts; check the design for possible inconsistencies; and automatically generate methodology diagrams and reports.

2.3.4 The AUML Methodology

AUML [20] is an extension to Unified Modeling Language (UML). AUML describes the mechanisms to model protocols for multiagent interaction. UML extensions include interaction diagrams, state diagrams and activity diagrams.

Interaction diagrams include sequence diagrams and collaboration diagrams. Interaction diagrams are used to define the behavior of groups of objects. State diagrams are used to model the behavior of a complete system. Activity diagrams are used to define courses of events for several objects and use cases.

2.3.5 The KAOS Methodology

The KAOS [18] methodology is a goal-oriented requirements engineering approach with a rich set of formal analysis techniques. KAOS stands for "Keep All Objects Satisfied" [18]. KAOS is

> "a multi-paradigm framework that allows combining different levels of expression and reasoning: semi-formal for modeling and structuring goals, qualitative for selection among the alternatives, and formal, when needed, for more accurate reasoning. Thus, the KAOS language combines semantic nets for conceptual modeling of goals, assumptions, agents, objects, and operations in the system, and linear-time temporal logic for the specification of goals and objects, as well as state-base specifications for operations." [23]

The KAOS ontology is composed objects (agents, entities, events and relationships), operations, goal and requisites, requirements and assumptions.

KAOS proposes three models: goal model, object model and operation model. G*oal model* represents the goals, and assigns to agents. O*bject model* is a UML model that refers to objects and their properties. O*peration model* defines various services to be provided by software agents.

2.3.6 Why Tropos?

The most important features that must be provided by the methodology within the scope of this work are support for early requirements phase, well defined analysis phases, and tool support for the methodology.

Gaia and KAOS methodologies are weak at the requirement analysis and detailed design phases. Prometheus methodology is similar to Gaia; however, as an advantage, it has good tool support. But, Prometheus methodology does not support the early requirement analysis phase. Tool support for Gaia and KAOS methodologies is not enough.

AUML can be used as a part of other methodologies. For example, AUML can be used in Tropos methodology at the Detailed Design Phase.

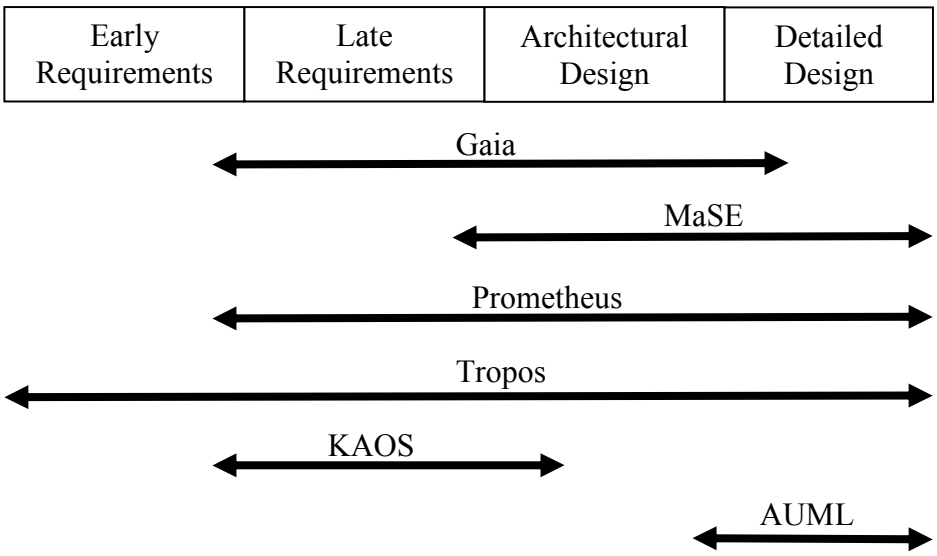


Figure 5. Coverage of Agent Oriented Software Development Methodologies. Adapted from [5].

The most important feature of the Tropos methodology is that it supports the software development process from early requirements to implementation.

Coverage of Agent Oriented Software Development Methodologies is illustrated in Figure 5. Therefore, Tropos methodology is complete and provides detailed descriptions. Also there is a good tool support, TAOM4E, for Tropos modeling. Tropos methodology can be combined with non-agent (e.g., object-oriented) software development techniques. For example, Tropos may be used for early development phases and UML or AUML can be used for later phases.

Another reason for choosing Tropos is that Tropos is an ongoing project. For the time this thesis is written, Tropos project is supported by a team from 6 different universities. See http://www.troposproject.org for the details of ongoing research, developed tools, industrial projects and Tropos related events.

Because we are mostly interested with the requirement analysis phase and modeling activities; we have preferred to use Tropos, which seems the most appropriate methodology with its support for early requirement analysis phase and with its modeling tool.

2.4 Command and Control

Command and control (C2), in the military concept, is defined by US DoD as

> "The exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission." [2]

NATO defines "command and control" as

> "the functions of commanders, staffs, and other command and control bodies in maintaining the combat readiness of their forces, preparing operations and directing troops in the performance of their tasks." [4]

On the other hand, most authors avoided to give a definition for C2.

> "Definitions of C2 are incomplete and potentially worthless unless a means is provided to measure existence (presence) or quality. The U.S. DoD definition of Command and Control provides the basis for a test that would indicate its existence. … The official DoD definition provides only one way to assess the quality of C2 and that is to equate the quality of C2 to mission accomplishment." [8]

> "In any rapidly evolving field (and Command and Control is certainly undergoing major changes in basic concepts and capabilities), definitions are problematic. … For a C2 definition to be useful to SAS-050, it needs to focus on why one does C2 and what functions an instantiation of C2 needs to accomplish to achieve its purposes." [4]

It is important to talk about the concept of Command and Control rather than definitions or usage of Command and Control. Command and Control are separate but interrelated functions [8]. Some distinctions have been defined between command and control. One distinction defined is the one that sees the Command as "art" and sees the Control as "science". Another proposed distinction underlines differences between the commander (command) and staff (control) [8].

Command and Control are fractal like concepts. C2 occurs at many levels of an organization from the enterprise level to mission level. C2, at the enterprise level, determines the purpose and capabilities of the organization. C2 at the mission level is about employing the "assets" of an organization to fulfill mission goals and objectives [4].

There are many different approaches to accomplishing Command and control functions. No specific approach or set of approaches defines what Command and Control means. According to [8], following functions are associated with Command and Control.

> "Establishing intent,
> Determining roles, responsibilities, and relationships
> Establishing rules and constraints
> Monitoring and assessing the situation and progress
> Inspiring, motivating, and engendering trust
> Training and education
> Provisioning" [8]

These functions are associated with mission or enterprise C2. They can be accomplished in very different ways. In this work, we mostly focus on "Monitoring and assessing the situation and progress" function as it is related to SA.

2.4.1 C2 Conceptual Model

We will use C2 conceptual model developed by Research and Technology Organization in NATO [4]. Conceptual Model is illustrated in Figure 6. This conceptual model is fractal like. Missions at different levels will have many concurrent, nested, and even overlapping instances of this model.

The domains of conceptual model are Information, C2 Approach, Sensemaking, and Actions, effects, and consequences. The C2 Approach establishes the conditions that affect information resources and processes. Sensemaking Process relies on Information Domain products.

Command and Control approaches in this conceptual model are related to Command and Control functions. According to [8]

> "How an enterprise chooses to accomplish the functions associated with Command and Control and the impacts and influences associated with the accomplishment of these functions need to be at the heart of a conceptual model of C2." [8]

Command Approach is a composite variable made up of *Allocation of Decision Rights, Patterns of Interaction Enabled, Information Distribution, Dynamics Across Purpose (Command)* and *Dynamics Across Time (Command)* [4].

Control Approach also includes *Restrictions on Decision Rights, Patterns of Interaction Not Allowed, Restrictions on Information Distribution, Dynamics Across Purpose (Control)* and *Dynamics Across Time (Control)* [4].

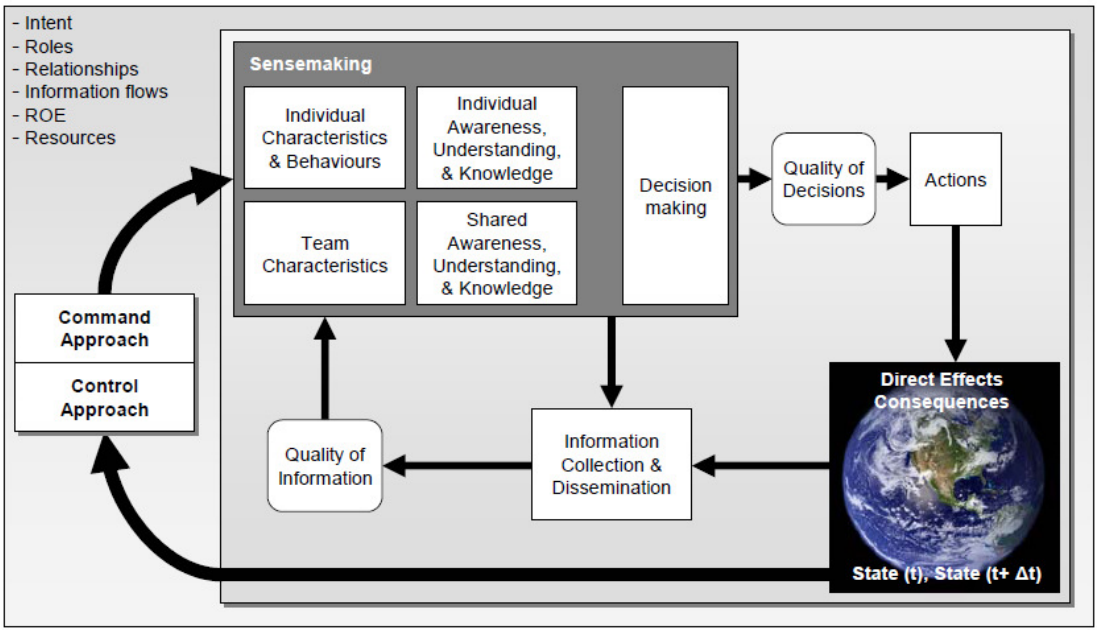


Figure 6. C2 conceptual model [4].

The information domain collects the information from environment. Sensors, direct or indirect, gather information about the situation. Attributes of sensors affects amount and quality of gathered information. Mobility, resolution, sensor coverage are the examples of sensor attributes [4]. Collected information must be stored in a data repository.

Uncertainties about the situation influence the availability of information. Ambiguity of situation, complexity of situation, uncertainty of situation, situational familiarity, and temporal focus, affect the information [4].

Sensemaking consists of a set of processes that begins with the perception of available information and ends with prior to taking actions [8]. Sensemaking process includes characteristics, behaviors, situation awareness and decision making. Collected information and quality of this information affects the Sensemaking process. The Sensemaking process involves both individual and

team sensemaking. The quality of decision making depends on the quality of shared understanding that is based on the shared awareness.

Shared awareness and understanding are developed in social processes of interaction among team members. Shared awareness and understanding depend on the quality of the awareness and understanding of the individual team members. Individual awareness and understanding are the result of cognitive processes in which available information is processed by individual team members. Both the social and the cognitive processes are shaped by the characteristics and behaviors of the team and its members [4].

Shared awareness and understanding result to a decision. Decision making is the "capability to form focused and timely decisions that proactively and accurately respond to these emerging opportunities and threats with available means and capabilities." [8]

Actions are executed after decision making process. Actions directly affect the environment. There may/should be consequences of these actions. The nature of the effects created by a particular action is a function of the action itself, when and under what conditions the action is taken, and the quality of the execution.

2.4.2 Situation Awareness

Before defining what the Situation Awareness (SA) is, it would be helpful to explain what SA is not. SA is not action. The understanding of a situation is different from the action taken in response to that situation. Second, SA is not long-term memory. Construction of SA is applicable only in dynamic situations where variables are changing. Third, the product of SA is not the same as the process of updating situation awareness. “It is important to distinguish the term situation awareness, as a state of knowledge, from the processes used to achieve that state” [11]. Thus, situation awareness is viewed as "a state of knowledge”.

Situation Awareness can be defined informally as “knowing what’s going on” and, formally, as

> “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future.” [11]

Model of SA is depicted in Figure 7.

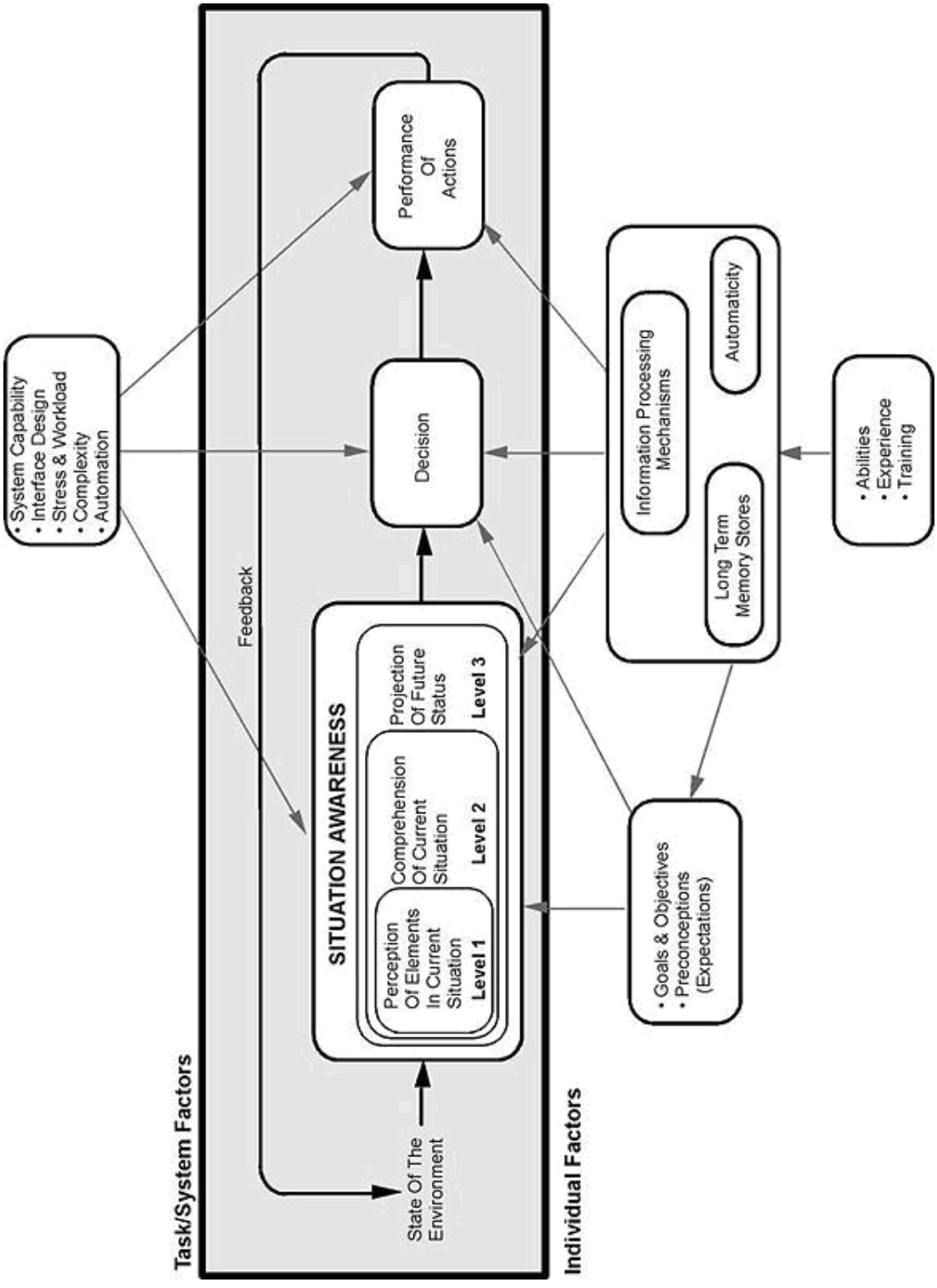


Figure 7. Model of situation awareness [11].

There are three steps of achieving SA: perception, comprehension, and projection [11].

Perception (Level 1 SA): The first step in achieving SA is to perceive the status, attributes, and dynamics of relevant elements in the environment. Level 1 SA involves the processes of monitoring, and simple recognition, which lead to an awareness of multiple situational elements (objects, events, people, systems, environmental factors) and their current states (locations, conditions, modes, actions).

Comprehension (Level 2 SA): This step is based on a synthesis of disjointed Level 1 SA elements through the processes of pattern recognition, interpretation, and evaluation. Level 2 SA requires integrating this information to understand how it will impact upon the individual's goals and objectives. This includes developing a comprehensive picture of the world, or of that portion of the world of concern to the individual.

Projection (Level 3 SA): The third level of SA involves the ability to project the future actions of the elements in the environment. Level 3 SA is achieved through knowledge of the status and dynamics of the elements and comprehension of the situation (Levels 1 and 2 SA), and then extrapolating this information forward in time to determine how it will affect future states of the operational environment.

Several variables can influence the development and maintenance of SA, These variables includes individual, task, and environmental factors. For example, individuals vary in their ability to acquire SA; thus, providing the same system and training will not ensure same SA across different individuals.

CHAPTER 3

ANALYSIS WITH TROPOS METHODOLOGY

3.1 The Scope of Analysis

The scope of this work is to model Situation Awareness in an Air Defense C2 System. The following paragraphs present the reasoning for this choice.

Command and Control are fractal-like concepts. C2 occurs at many levels of an organization from the enterprise level to mission level. We preferred to model C2 at a level between enterprise level and mission level. If we had chosen to model at mission level, model would include specific details to that mission. In that case, model would not represent the general C2 capabilities. If we had chosen to model at enterprise level, model would be more general or at the strategic level that we would not able represent some C2 capabilities.

In the military scope, C2 can be at strategic level, operational level and tactical level. According to [2], at the strategic level, a nation determines national strategic security objectives, and develops and uses national resources to achieve these objectives. At operational level, major operations are planned, conducted, and sustained to achieve strategic objectives within operational areas. Activities at this level link tactics and strategy. At the tactical level, battles and engagements are planned and executed to achieve military objectives assigned to tactical units or task forces [2]. We preferred to model a C2 system at the tactical level.

C2 systems may have similarities in general but may differ in details. C2 conceptual model is sufficient to model C2 systems in general with basic

operations, i.e. command, control, sensemaking, actions. But, when trying to model C2 in more details, a specific type of C2 system must be chosen, because each system has different features. For example, C2 systems may or may not contain certain types of sensors. Sensors also may differ among systems. There may be radars, environment sensors (heat, pressure, and air condition sensors) or video cameras. To be able to model a C2 system, we had to choose a specific one. Thus we have chosen air defense system.

Modeling the whole air defense system at architecture level would cause the model to be unmanageably large, hence, we could not afford to model towards architectural design phase. To keep the model understandable and manageable, it was necessary to focus to some parts of air defense system. Therefore, we decided to keep the scope of this work as situation awareness in air defense system. Situation awareness is chosen, because it is prevalent in all C2 systems.

Basic objective of the air defense system is to defense a boundary, an area, or a critical position against air tracks. Tactical Picture helps the operator (i.e. officer in charge) to establish situation awareness. Tactical Picture, which is typically displayed symbolically on a geographic display, provides basically air track information, unit information which can be own unit or enemy unit, and battlefield geometries required for air defense operations. Tactical Picture plays role in the first level of situation awareness; perception of elements in the environment. The second and third levels of SA, comprehension and projection, are operator related operations that operator is responsible to comprehend those elements and predict the states of those elements in the near future.

As a result, scope of this work is situation awareness in a C2 system, namely, air defense system, especially Tactical Picture component which provides input for the first level of situation awareness activities.

3.2 Early Requirements

At the early requirements phase, we will first identify actors and relations among them. Then, we will focus on each actor to find their goals, plans, etc.

3.2.1 Actor and Dependency Modeling

Although we focus on SA analysis, all C2 fundamental activities are related to each other. To present the context of SA, other activities are defined as simple goals i.e. goals that are not decomposed any further. SA requirements are analyzed in detail.

We begin C2 conceptual modeling by identifying actors. Conceptual model, depicted in Figure 6, includes Information domain, C2 Approach, Sensemaking, and Actions, effects, and consequences.

First we begin with modeling of the Command Approach and Control Approach. In the concept of air defense units, Commander owns the both Command and Control functions. Therefore, we define *Commander* Actor as the main actor of C2. Command and Control, and Sensemaking activities of C2 are related to Commander.

Information must be provided to *Commander* Actor for Sensemaking activities. Tactical Picture helps Commander (i.e. officer in charge) to establish situation awareness and then to make decisions. *Tactical Picture Provider* is defined another actor that supplies information to Commander. *Tactical Picture Provider,* which displays Tactical Picture on a map, helps Commander to percept the elements of the environment. This actor is also responsible for maintaining the tactical picture by collected information from various sources. Providing tactical picture activity is modeled as *Present Tactical Picture* hard goal under *Tactical Picture Provider* actor. Quality of tactical picture affects the quality of SA. Increasing the quality of tactical picture is a requirement for *Tactical Picture*

Provider actor. Since the increasing the quality is a non-functional requirement, it is defined as soft goal, *Increase Quality of Tactical Picture*. As a result, *Commander* actor depends on *Tactical Picture Provider* actor's *Present Tactical Picture* hard goal and *Increase Quality of Tactical Picture* soft goal.

Actions activity must be fulfilled by an actor called *Subordinate*. Here, we include *Subordinate* actor in a sense of a representative of a group. Subordinate actor does not have to correspond to a single entity but a group of entities. Actor that does *Actions* activity can vary for different C2 systems. Subordinate performs actions and reports the results. Subordinates also collect information from environment and report them to their related units.

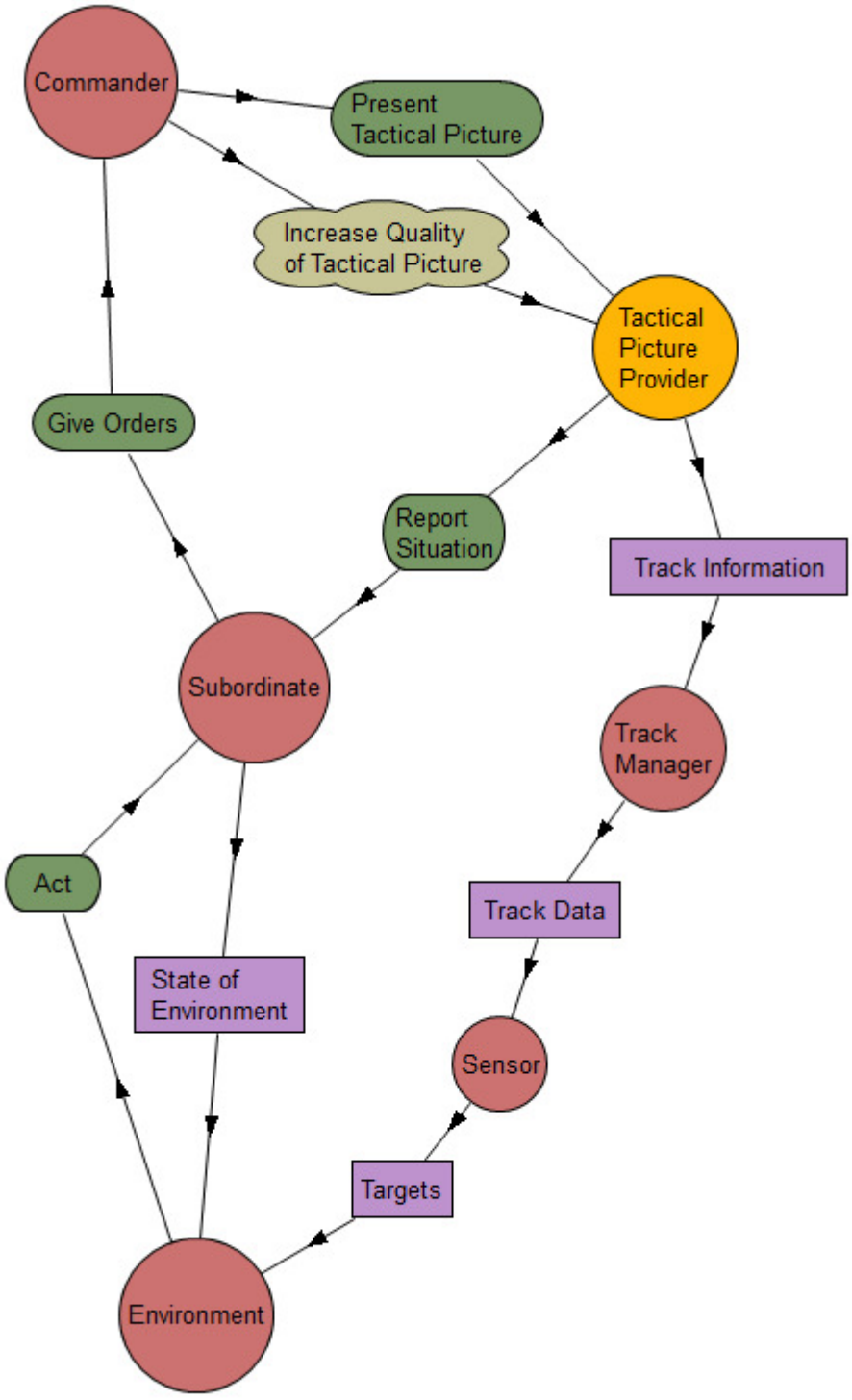


Figure 8. C2 early requirements actor diagram.

There are two alternative ways of modeling dependencies between *Commander* and *Subordinate* actors. *Subordinate* actor has *Act* goal and *Commander* actor has *Give Orders* goal. *Act* goal is dependent to *Give Orders* goal. We would show dependency as the way that *Commander* actor depends on *Subordinate* actor's *Act* goal. This representation shows that *Commander* actor uses *Subordinate* actor's *Act* goal to achieve its goal. As an alternative, we could show dependency as the way that *Subordinate* actor depends on *Commander* actor's *Give Orders* goal. This representation shows that *Subordinate* actor depends on, or affected by *Give Orders* goal. When the orders are given, *Subordinate* actor has to achieve its *Act* goal. We preferred to use the second way.

Actions activity has direct effects and consequences on the environment. We define a new actor, *Environment*, in order to show other actors' relations with environment. Environment is defined as an external actor for the C2 system. By external, we mean that environment is not a part of the system. Environment includes geographical information, weather conditions and battlefield information (enemy and own forces information, target information).

Relation between *Environment* actor and *Subordinate* actor is modeled as the following. *Subordinate* actor depends on the state of environment which is defined as resource under *Environment* actor. *Environment* actor depends on *Act* goal of *Subordinate* actor. There may be different ways of modeling this relation; but these two relations are enough to illustrate the concept.

We modeled one part of Information domain of Conceptual Model by defining *Subordinate* actor as an information source. In an air defense system, track information must be presented in tactical picture. Sources of track information are own radars and external track information through TDL i.e. Link-11, Link-16. More generally, in a C2 system, information is received from its sensors and presented in tactical picture. For the rest of the work, we will to use a general term, sensor, instead of radar. Here, we use the term, sensor, in a sense of a

representative. Sensor may be radars, environment sensors (heat, pressure, and air condition sensors), video cameras, or external information sources such as TDL, possibly with their (human) operators. We define a new actor, *Sensor*, to represent sensors. Sensor actor depends on *Environment* actor by targets.

Sensor actor does not have to be one actor. There may be, or should be more than one sensor in a C2 system. Track data are obtained by different sensors. Tactical picture information must be produced form sensor data. In other words, information must be derived from data which is the lowest level of abstraction. Later knowledge, in other terms situation awareness, is derived from this information by user. To produce track information, sensor data must be identified, and tracked. Also, track data from different sensors must be combined. We define *Track Manager* Actor to be responsible for post processing activities after detection phase. *Track Manager* Actor depends on *Sensor* Actor by its track data. *Tactical Picture Provider* actor depends on *Track Manager* actor's *Track Information* resource.

*Track Manager A*ctor could be thought and modeled as part of *Tactical Picture Provider* actor. There is no restriction to do that. Here, we preferred to model sensor processing activities separately.

Actors and dependencies with each other are illustrated in Figure 8.

In our model, some actors model real human beings, while others model software (running on special or general purpose hardware). Commander, in general, is a human being, possibly with his staff. Software may be used to achieve some commander's goals, like Take Mission, Give Orders, etc. In our model, *Commander* actor may represent a man-machine system. For the details, the *Commander* actor must be analyzed at the late requirement analysis phase, but this point is not the focus of the present model. The *Tactical Picture Provider* actor represents an operator and a subsystem or component that provides both

tactical picture and a user interface. Operator (i.e. officer in charge) that follows tactical picture is responsible to present intended tactical picture to Commander. User Interface gets user (Commander) input; asks for user choice and gets orders from the user. Subordinate, in general, is a human being. In our model, S*ubordinate* actor represents a man-machine system. Software part of S*ubordinate* actor helps subordinate to communicate with *Commander* and *Tactical Picture Provider* actors, and may help subordinate to achieve his *Act* goal. *Sensor* actor represents a sensor with both software and hardware. *Track Manager* actor includes both hardware and software components. *Track Manager* actor may also represent a man-machine system that operator may take role in identification, or data fusion. Environment actor represents the real environment; geographical information, weather conditions and battlefield information.

3.2.2 Goal Modeling

The *Commander* actor is at the center of Command and Control process. *Command Approach, Control Approach* and *Sensemaking* activities are performed by this actor. Other actors help the *Commander* actor to accomplish his intent. Since we will not go into details of *Commander* actor in this thesis, here essential goals of *Commander* actor are given. *Commander* has intent (the objective or goal). Aiming to achieve intent can be defined as a hard goal.

Intent, formally, is defined as

> "a concise expression of the purpose of the operation and the desired end state that serves as the initial impetus for the planning process. It may also include the commander's assessment of the adversary commander's intent and an assessment of where and how much risk is acceptable during the operation." [2]

To achieve the intent, firstly, commander establishes intent. Intent may be determined by a higher authority. There are cases where there is no higher authority that can, in practice, determine the intent. In that case, intent is determined by the Commander. Here we assume, Commander takes mission from some higher authority; and according to that mission, establishes intent. Commander analyzes the mission after taking it. Results of analysis help Commander to establish intent. In fact, intent would be related to completion of the mission. *Take Mission* and *Analyze Mission* are defined as hard goals under *Achieve Intent* goal with AND decomposition.

To achieve the intent, *Commander* has to be aware of current situation. Therefore *Be Aware of Situation* is defined as a hard goal under *Achieve Intent* goal with AND decomposition. *Commander* actor needs to get information to form SA. *Commander* actor depends on *Tactical Picture Provider* actor's *Present Tactical Picture* goal.

By using information in Tactical Picture, *Commander* perceives the elements of current situation. After reviewing the available information, Commander comprehends current situation and projects it to near future. These SA activities are defined as hard goals under *Be Aware of Situation* goal and combined with AND Decomposition. Increasing the quality of SA increases the quality of decisions, therefore helps to achieve intent. Increasing the quality of SA can be defined an objective of Commander. Since it is difficult, in general, to put forth clear-cut separation criteria between acceptable and unacceptable levels of SA quality, *Increase Quality of Situation Awareness* is modeled as a soft goal. Contribution of SA to quality of SA, modeled as contribution relation.

The Decision Making activity follows the Situation Awareness process. Therefore, this activity is modeled as *Make Decision* hard goal under *Achieve Intent* goal with AND decomposition. Quality of decision is influenced by current situation, intent, mission and decision making process. Increasing the quality of decision can be modeled as a soft goal, *Increase Quality of Decision*. *Make Decision* goal contributes to this soft goal. Variables characterizing the quality of decisions are analyzed in Chapter 8 of NATO Report [4]. High quality of SA increases the quality of decision; therefore *Increase Quality of Situation Awareness* soft goal contributes to *Increase Quality of Decision* soft goal. As the present work is not interested in the details of Decision Making process, *Make Decision* is left as a simple (i.e. un-decomposed) goal. In the same way, other *Commander* actor activities such as Take Mission, Analyze Mission and Determine Roles, Responsibilities and Relationships are not analyzed in detail.

Command activities, Take Mission, Analyze Mission, Make Decision, Determine Roles, Responsibilities/Relationships and Give Orders, are to be done in mentioned order. In Tropos, we cannot explicitly indicate the sequence of elements. This issue will be discussed later. Command activities are modeled as hard goals under *Achieve Intent* goal with AND decomposition.

Following the process of SA and Decision Making, Commander determines subordinates' roles and responsibilities. Once actions are decided, Subordinate takes over its responsibilities and fulfills them. Subordinate is dependent to Commander through its goal *Give Orders*. In this paper, *Give Orders* goal defines the communication between Commander and action performing entities, which we call Subordinates.

Environment is affected by Subordinates' *Act* goal. This relation is shown with hard goal dependency between *Environment* actor and *Subordinate* actor via *Act* goal. Environment is in some state at a given time and its state changes dynamically. Environment's state can be modeled as a collection of information. Therefore, state of environment is modeled as a resource, *State of Environment*. The Subordinate actor depends on E*nvironment* actor's *State of Environment* resource.

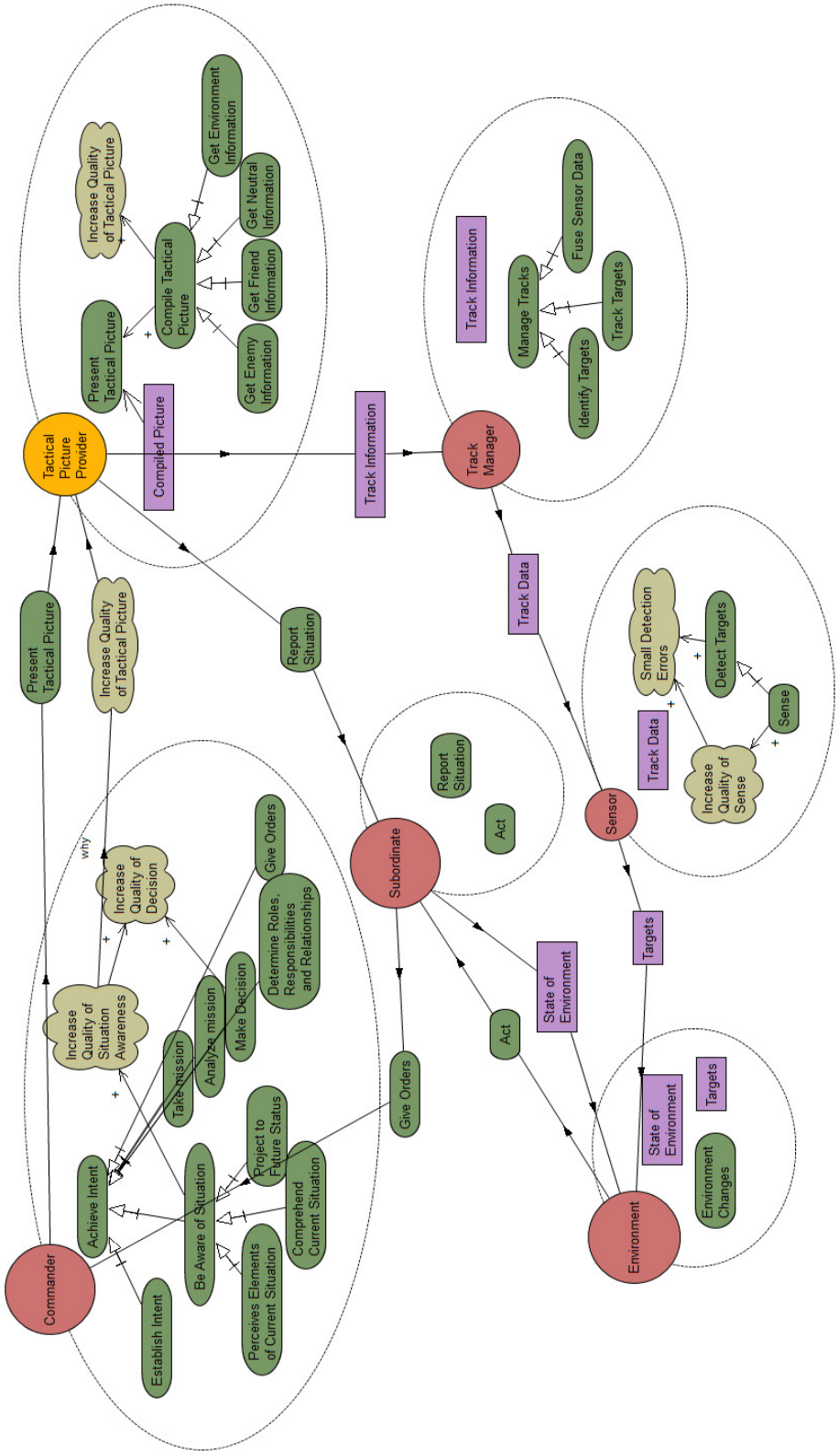


Figure 9. C2 early requirements goal diagram.

Different sensor types produce different types of data; plot data and track data. Here, we model sensor as producing track data. *Sensor* detects targets, or in other words, gets environment information. *Sensor* actor depends *Environment* actor's *Targets* resource. *Detect Targets* is defined as a hard goal under *Sensor* actor. *Detect Target* goal represents the activities of producing track data from plots and classification. Minimizing the detection errors must be a goal of *Sensor* actor. Since minimizing detection errors is a non-functional requirement; *Small Detection Errors* is defined as soft goal. To achieve *Detect Targets* goal, sensor must sense the environment. Therefore *Sense* is defined as hard goal under *Detect Targets* goal with AND decomposition. Increasing the quality of sense must be a goal of *Sensor* actor. Since, increasing the quality is a non-functional requirement; *Increase Quality of Sense* is defined as a soft goal. High quality of sense activity decreases the detection errors, therefore we add a contribution relation from *Increase Quality of Sense* goal to *Small Detection Errors* goal.

Track data produced by *Sensor* actor is modeled as resource, *Track Data*. Track data will be used by *Track Manager* actor. This situation is modeled as *Track Manager* actor depends on *Sensor* actor's *Track Data* resource.

Track Manager actor is responsible for managing tracks. We define *Manage Tracks* as a hard goal of *Track Manager* actor. This actor does the post processing activities after detection phase. These activities include identification, tracking and sensor fusion which are defined as hard goals, under *Manage Tracks* goal with AND decomposition: *Identify Targets*, *Track Targets*, and *Fuse Sensor Data*. Here, we assume that classification activity is achieved at sensors.

Identification is defined as the process of determining the friendly or hostile character of an unknown detected contact [2]. Tracking is defined as precise and continuous position-finding of targets by radar, optical, or other means [2].

Sensor Fusion is the process of combining multiple elements of data from different sensors in order to produce information of tactical value to operator. Sensor fusion reduces the information load on operators and improves the tactical picture quality. Sometimes, more abstract term Data Fusion is used instead of Sensor Fusion. Sensor Fusion may occur either at the plot level or at the track level. Since we modeled sensor as producing track data instead of plot data, Sensor Fusion activity is modeled at the track level.

Track Manager actor produces track information from track data. Track information is modeled as a resource, *Track Information*. Track information will be used in Tactical Picture; therefore there exists a dependency relation from *Tactical Picture Provider* actor to *Track Manager* actor's *Track Information* resource.

A tactical picture represents the immediate environment of interest to commander. It assists commander in forming a mental representation of the world and guides the commander in taking the most appropriate actions. To be able to compile the tactical picture, *Tactical Picture Provider* actor collects situational information. Information comes from different sources such as Subordinates and processed information from Sensors. Tactical picture actor's main goal is to *Compile Tactical Picture*. Tactical picture is composed by putting together available information about enemy, allies, neutral entities and environment. These are sub goals of *Compile Tactical Picture* goal combined with AND decomposition; *Get Enemy Information, Get Friend Information, Get Neutral Information*, and *Get Environment Information*. In the Late Requirement Analysis phase, we tactical picture information will be analyzed in detail; here, information is grouped according to their identity, friend, hostile, and neutral. Information at this phase may be grouped as track information, unit picture etc.

Quality of information affects the quality of tactical picture, therefore quality of SA. Increasing the quality of information must be a goal of *Tactical Picture*

Provider actor. Since increasing the quality is a non-functional requirement, it is defined as soft goal, *Increase Quality of Information*.

Goal Diagram of Early Requirements Phase of Situation Awareness is depicted in Figure 9.

Communication patterns among actors can be complicated; they need to be analyzed in detail, in terms of communication protocols, standard formatted messages, and so on. During the late requirement analysis and architectural design, new actors and hard/soft goals must be added to define inter-actor communication.

3.3 Late Requirements

3.3.1 Actor and Dependency Modeling

Since we focus on SA, *Tactical Picture Provider* actor will be analyzed in the Late Requirement Phase. While constructing actor model of *Tactical Picture Provider* at the late requirement phase, we again focus on SA. Therefore, while defining actors, actors that are not directly related to SA are not studied in detail. In real systems, there may be much more actors than defined here. Also relations between actors should be much more complicated in real systems.

Tactical Picture Provider actor mainly collects information and compiles tactical picture. Information may come from subordinate and sensors. Here, we introduce a new actor, *Communication*. *Communication* actor sends messages to and receives messages from *Subordinate* and *Interpreter* actors.

Collected information must be managed and stored in a data repository. *Data Repository Manager* and *Data Repository* actors perform data management and data storage. *Data Repository Manager* actor process received messages and manages data in data repository. This actor basically provides data flow between data repository and the rest of actors.

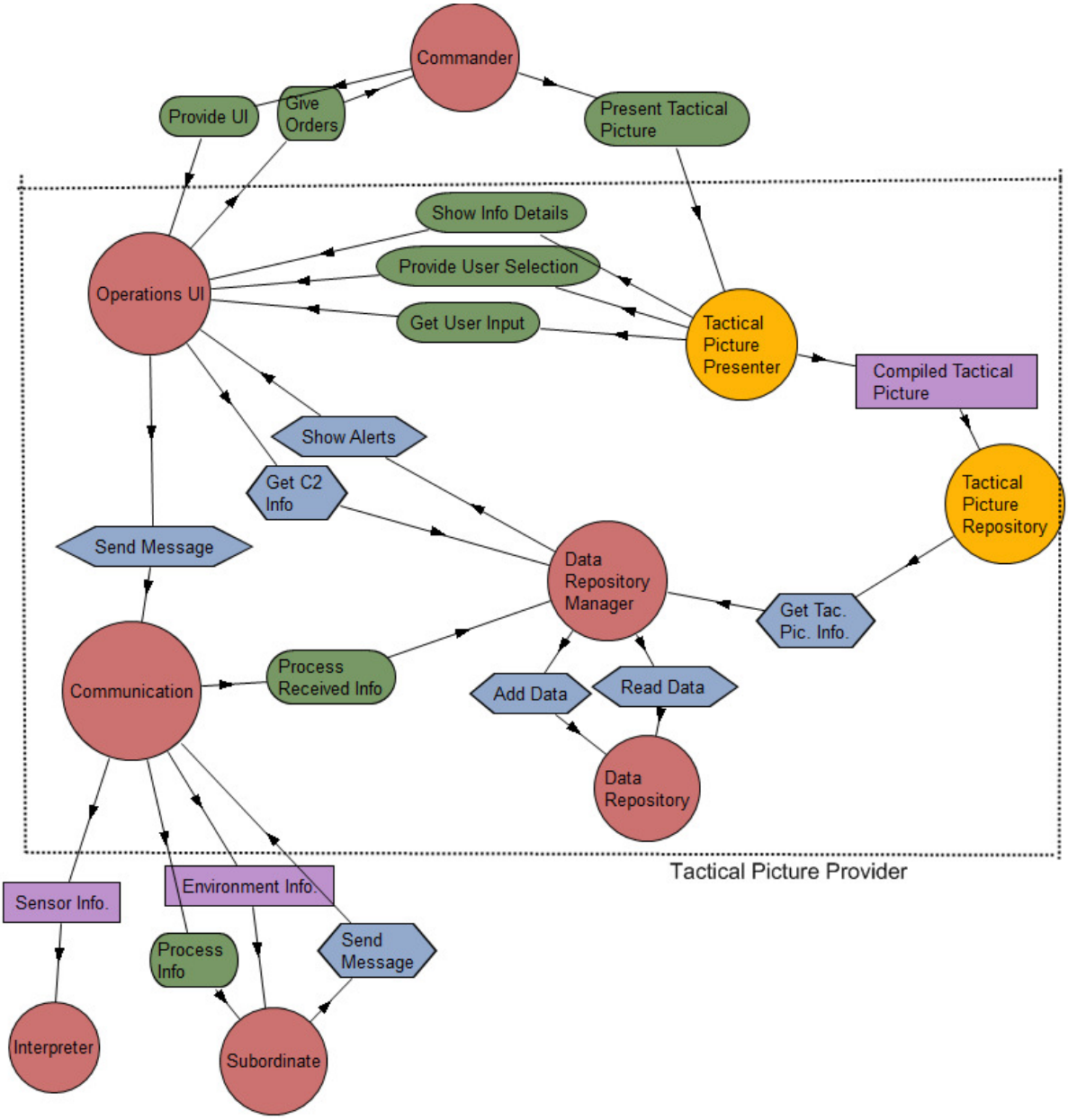


Figure 10. C2 SA late requirements actor diagram.

Tactical Picture Provider goals, *Compile Tactical Picture* and *Present Tactical Picture*, lead us to introduce new actors as *Tactical Picture Repository* and *Tactical Picture Presenter* respectively. *Tactical Picture Repository* actor basically compiles the tactical picture information by collecting the information

and then storing them. *Tactical Picture Presenter* actor represents tactical picture by using Compiled Tactical Picture produced by *Tactical Picture Repository*.

Finally, there is an actor for user operations. This actor, called *Operations UI*, provides communication with *Commander* actor. Operation UI basically actor shows information to user and gets commander orders.

Tactical Picture Presenter actor can be thought as a GIS application. *Operations UI* actor can be thought as an application that contains GIS application and provides additional user interfaces.

Actor Diagram of SA at the Late Requirement Analysis Phase is presented in Figure 10.

Because the scope of this thesis is defined as the SA in C2, we will further go into details of *Tactical Picture Repository* and *Tactical Picture Presenter* actors. Other actors and their relations will not be analyzed in details. In real systems, there must be more actors and actor relations, making them more complex. For example, in real systems, there should be a specialized actor for planning operations.

3.3.2 Goal Modeling

Goal Diagrams of *Tactical Picture Repository* and *Tactical Picture Presenter* actors will be studied in this section. *Tactical Picture Presenter* actor depends on the resource, *Compiled Tactical Picture,* produced by *Tactical Picture Repository*. Goal Diagram is depicted in Figure 11.

The main goal of *Tactical Picture Repository* actor is to *Compile Tactical Picture*. To achieve this goal, it gets information and stores them. Information stored in Data Repository can be reached from *Data Repository Manager*. Therefore, *Tactical Picture Repository* depends on *Data Repository Manager* by *Get Tactical Picture Info* plan. Details of *c*ommunication between *Data Repository Manager* and *Tactical Picture Repository* will be analyzed in Architectural Design phase.

In an air defense system, weather conditions affect radar performance, air track movement capabilities and effectiveness of weapon systems. For example, bad weather conditions decreases visibility of air tracks; therefore decrease the effectiveness of weapon systems. Weather conditions may also affect the movement capabilities of air defense systems. Geographical information like barricades, foundations or events positions is helpful for air defense systems, because they affect the positioning of air defense systems. Other geographical information types can be utilized. Weather and geographical information help to construct situation awareness. These are added to our model under goal *Environment Information*.

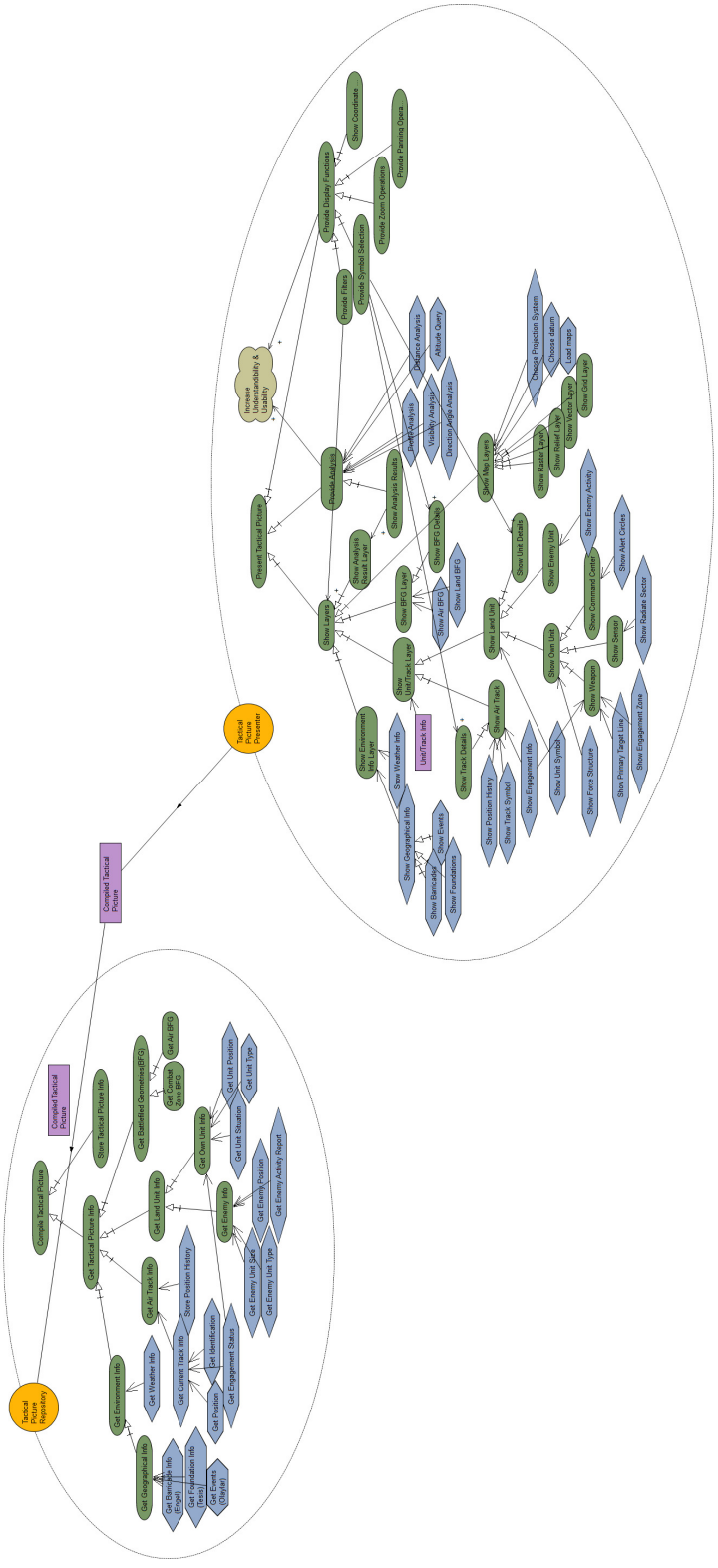


Figure 11. C2 SA late requirements goal diagram.

Air track information is needed for SA. At least track's position, identification and engagement status must be known. Position data must include coordinates and altitude. We might be included much more information about air tracks, such as IFF code, speed, etc.; but for the sake of simplicity, we preferred to keep model more simple and understandable.

Land units' information must also be known by Commander. Land units may be own forces or enemy forces. Own forces may be air defense units or one of other types. Own forces position, situation and type information, and enemy forces' positions are the necessary information to understand the situation. Commander may want to move air defense units to be able to prevent them from enemy forces. In the battlefield, there may be much more complex situations; land units information helps air defense commander to understand the current situation. Land unit information added to model as hard goals and plans.

Battlefield geometries (BfG) model battlefield. BfGs can be two types. First, combat zone BfGs models ground. Mine fields are the example for combat zone BfG. Second, air BfGs model the airspace. There are hundreds types of BfGs. In this thesis, we will not go into details of BfGs.

As a summary, information needed to construct Tactical Picture is composed of environment, air track, land unit, and BfG information. Elements of *Tactical Picture Repository* actor is depicted in Figure 12.

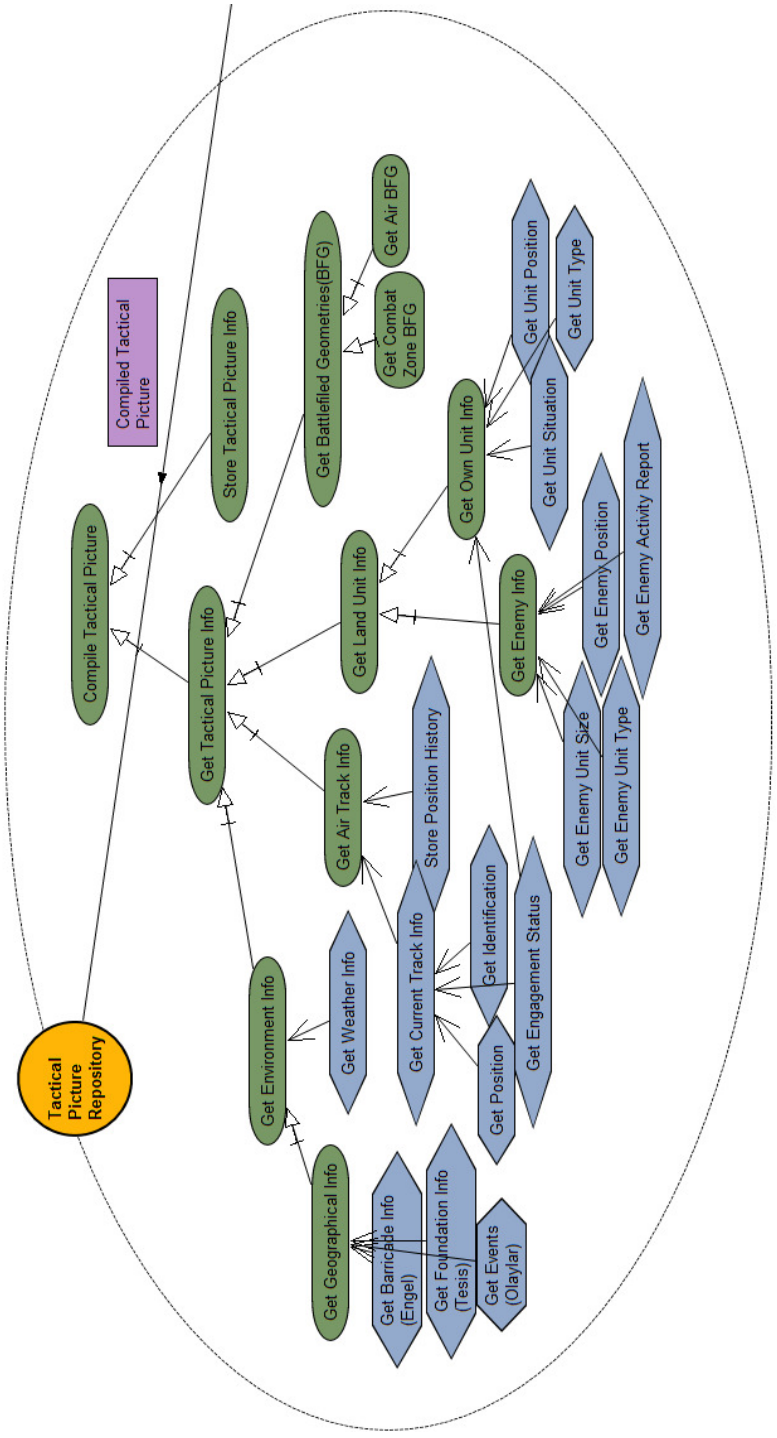


Figure 12. Goal diagram of *Tactical Picture Repository* actor.

Tactical Picture Presenter actor represents the GIS application that presents information on a map. *Tactical Picture Presenter* actor depends on the resource, *Compiled Tactical Picture,* produced by *Tactical Picture Repository*. The main goal of *Tactical Picture Presenter* actor is to "Present Tactical Picture". Compiled tactical picture information can be grouped as layers. Each layer contains similar kind of information.

Environment Layer shows weather and geographical information. Barricades, Foundations, Bridges and Events have different symbols according to their types and geometric types. Geometric types may be point, circle, polygon, line or volume. According to geometry type, drawing rule changes. These entities may be hostile or friendly. This affiliation affects the presentation. For example, hostile/friendly types might be drawn in red/blue, or vice versa. These items can be planned or reported. Planned items may be drawn with dashed lines, while reported items may be drawn with solid lines. These items may have start and end time. There may be many other features. *Tactical Picture Presenter* actor must present this information according to its details. To be able to keep the model simple, we did not add these details to model.

Unit/Track layer contains both unit and track information. Land units can be hostile units, friendly units and own units. Each unit has detailed information just like geographical items; we will not go into details. Each unit has specific drawing rules according to its details. Own unit must be drawn with its force structure, i.e. superior and sub units. If own unit is a weapon, primary target line and engagement zone must be shown in the tactical picture. If own unit is a sensor, its radiate sector must be shown. If own unit is a command center, alert circles must be shown.

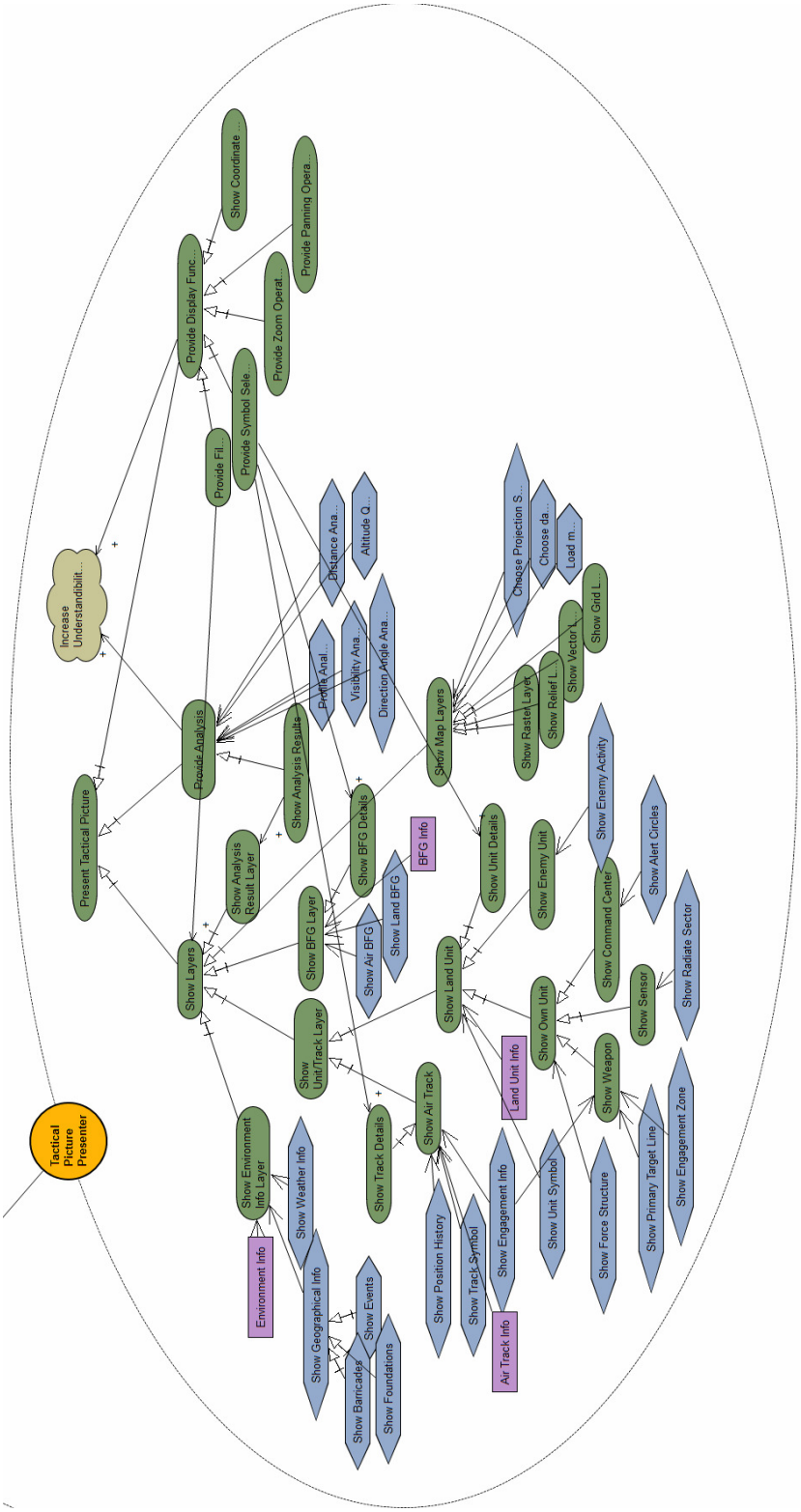


Figure 13. Goal diagram of *Tactical Picture Presenter* actor.

Air tracks must also be placed in the tactical picture. Tracks are drawn with different symbols according to their identification and types. Possible track types are rotary wing, fixed wing unmanned air vehicle or missile. In the tactical picture, track positions must be updated frequently. Showing track history improves situation awareness. If the track is engaged to an own unit, this relation must be shown in tactical picture.

Battlefield geometries have two kinds: Air BfG (or Air Control Order) and Land BfG. BfGs may have different geometry type, as point, circle, polygon, line, polygon volume, etc. BfGs are important for commander when controlling the battle field. We will not go into details of BfGs in this work.

Tactical picture elements must be drawn on a map to be able to illustrate position of elements. Therefore, *Tactical Picture Presenter* actor has a hard goal "Show Map Layer". Maps can be type of Raster, Relief, Vector and Grid. Each map type can be selected to be shown or not. "Show Map Layer" hard goal has sub hard goals as "Show Raster Layer", "Show Relief Layer", "Show Vector Layer" and "Show Grid Layer". Projection System and Datum should be selectable by user. Because, selecting projection system and datum are defined activities; they are modeled as plans.

Tactical Picture has a lot of information, therefore filter must be provided to increase usability and understandability. Filter may be applied to layer or types. For example, user may want to filter map raster layer or BfG layer. User also may want to filter some types of BfGs or some types of units.

Tactical Picture Presenter actor must provide display functions to increase usability and understandability of tactical picture. Zoom and panning operations may help user to understand the current situation. User may also want to see the details of an item. *Tactical Picture Presenter* actor has a hard goal "Provide Symbol Selection" for this purpose. When a symbol is selected, details of symbol

are displayed. When cursor moves around tactical picture, current position information may be helpful to commander. “Show Coordinate Info” hard goal is defined for this purpose.

Tactical Picture Presenter actor must provide analysis functions to increase usability of tactical picture. There may be several types of analysis, 5 of them are defined in our model. Making an analysis is defined activity, therefore providing analysis functions are modeled as plans. Distance Analysis, Altitude Query and Direction Angle Analysis have single results. Visibility Analysis results must be shown on map. Therefore, we need to add an extra layer to Tactical Picture, “Analysis Result Layer”. Analysis results may be shown in a separate window or in tactical picture.

Elements of *Tactical Picture Presenter* actor is depicted in Figure 13.

3.4 Architectural Design

The Architectural Design phase focuses on the system specification. Architectural Design defines the architecture of the system in terms of sub-systems and their relations. In the model, sub-systems are represented as actors and relations are represented as dependencies.

Architectural Design phase consists of three steps. Firstly, overall architectural organization is defined. Sub-actors are introduced in the system. New actors may be added to model during goal analysis and after the choice of a specific architectural style. Dependency, goal and plan modeling activities are done for new actors. As a result of architectural design, extended actor diagram including new actors and their dependencies with the other actors are presented. The final result of this first step is an extended actor diagram that presents new actors and their dependencies with the other actors [5].

Second step consist of identifying capabilities needed by the actors to fulfill their goals and plans. Capabilities are identified by analyzing the extended actor diagram. Finally, the last step consists of defining a set of agent types and assigning each of them one or more different capabilities [5].

At the architectural design phase of our model, we will focus on actors *Tactical Picture Repository* and *Tactical Picture Presenter*. First, new actors will be introduced. Then, we will identify the capabilities of actors needed by the actors to fulfill their goals and plans. We will identify capabilities by extending the actor diagrams.

3.4.1 Step 1: Defining Sub Actors and Goals

3.4.1.1 Tactical Picture Repository Actor

We introduce 3 new actors under *Tactical Picture Repository* actor at this phase: *Information Provider*, *Information Analyzer* and *Tactical Picture Storage Manager*. Actor diagram of *Tactical Picture Repository* is illustrated in Figure 14.

Information Provider actor gets the information from *Data Repository*. This actor reads the information and synthesizes results for the use of *Tactical Picture Repository*. *Tactical Picture Repository* depends on *Information Provider* actor by *Get Info*rmation plan. *Get Information* is defined as a plan, because getting information involves a well defined procedure.

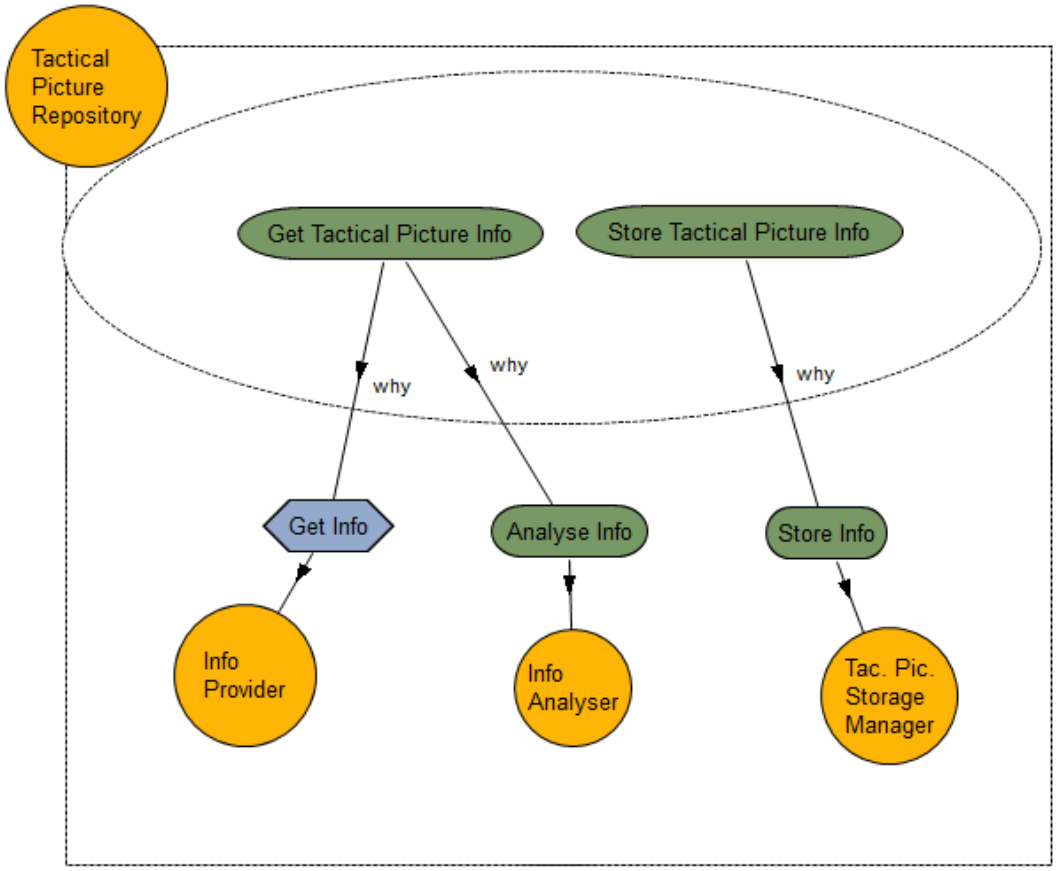


Figure 14. Actor diagram of *Tactical Picture Repository* actor.

Get Information plan has two sub plans to achieve its goal: *Read Information* and *Synthesize Results*. *Read Information* plan reads Tactical Picture Information from *Data Repository*. It reads unit information, environment information and battlefield geometry information. To do this, it depends on some capabilities of *Data Repository Manager* actor: Read Unit Information, Read Environment Information and Read BfG Information.

After reading the information, *Information Provider* actor synthesizes the results by its plan defined as *Synthesize Results*. *Synthesize Results* plan depends on information which defined as resource, *Query Results*, in *Data Repository*

Manager actor. Goal Diagram of *Information Provider* actor is depicted in Figure 15.

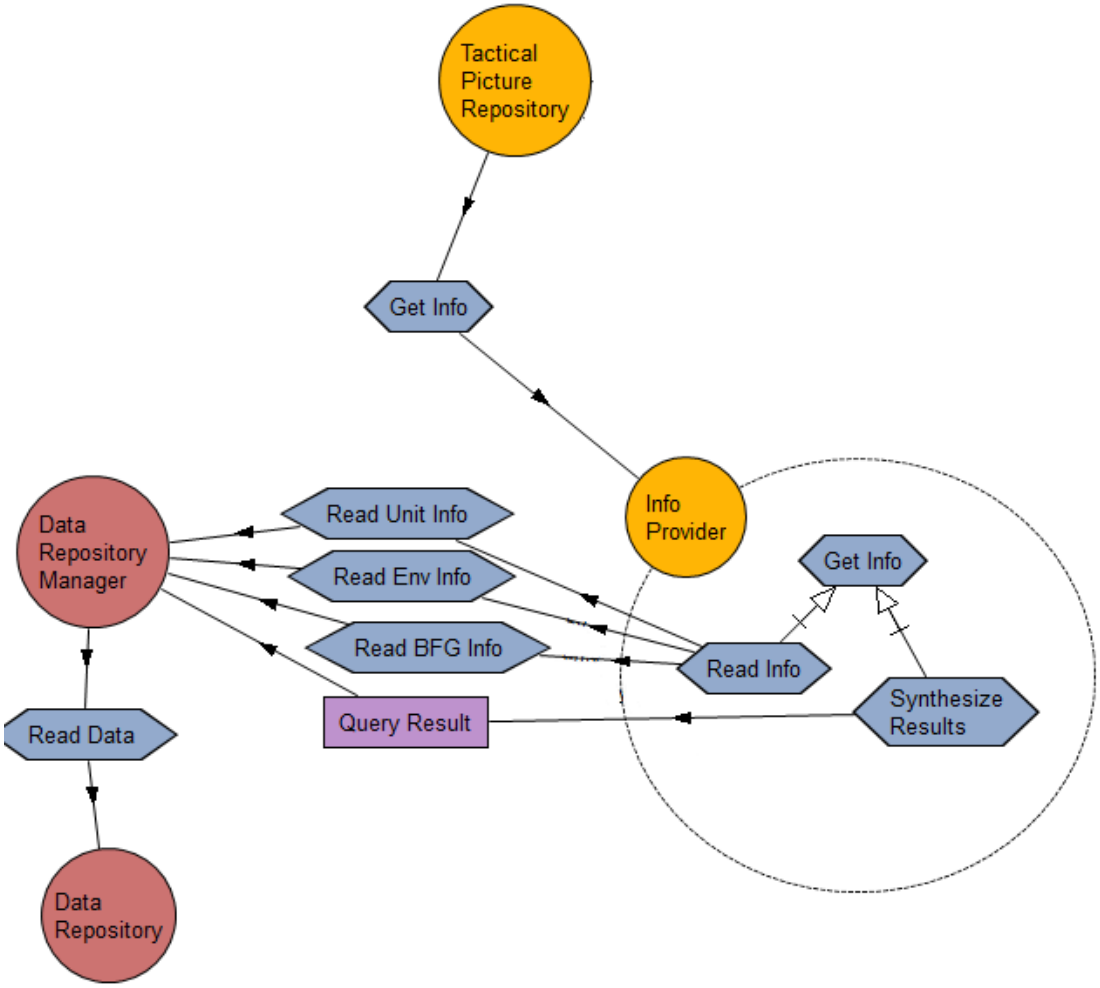


Figure 15. Goal diagram of *Information Provider* actor.

Main goal of *Information Analyzer* actor is to *Analyze Information*. Information read from Data Repository contains some irrelevant information about tactical picture. For example, some of unit details are not needed in tactical picture. This information is needed when symbol is selected to see the details. Some information may also be outside of scope of tactical picture. Scope of tactical picture is a wider area than the visible area. Tactical picture provides zoom in

and zoom out operations. Scope can be defined as the area when zoom out is at maximum level.

Unneeded information must be removed. To achieve this, we define sub goals and plans in *Information Analyzer* actor. AND decomposed sub goals are *Analyze by Area, Analyze by BfG, and Analyze by Unit*. Two plans help actor to achieve its goals: *Remove Information Outside of Scope*, and *Remove Irrelevant Unit Information*. Analyzed info is defined as a resource, *Analyzed Information*.

Tactical Picture Storage Manager actor depends on *Information Analyzer* actor by *Analyzed Information* resource. *Tactical Picture Repository* actor depends on *Information Analyzer* actor by *Analyze Information* goal. Goal Diagram of *Information Analyzer* actor is depicted in Figure 16.

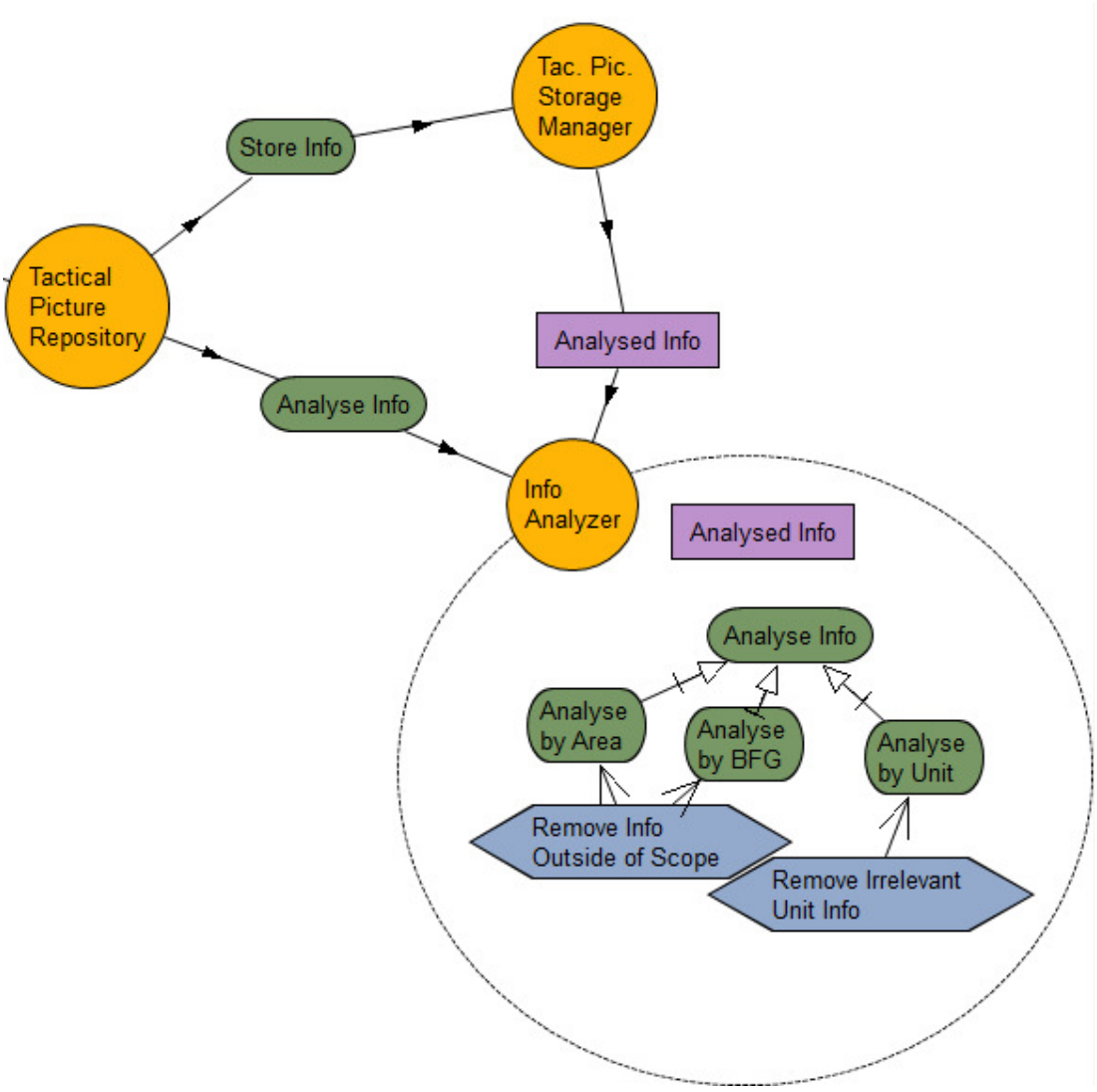


Figure 16. Goal diagram of *Information Analyzer* actor.

Tactical Picture Storage Manager is the last actor defined under *Tactical Picture Repository* actor at the architectural design phase. The primary goal of this actor is to store information. To achieve this, it puts information in Tactical Picture Repository. Putting information to somewhere is a well defined activity, therefore identified as a plan, *Put in Tactical Picture Repository*. This actor should be more complicated at the detailed design phase, but at this phase, this detail level is enough to illustrate the concept. Goal Diagram of *Tactical Picture Storage Manager* is presented in Figure 17.

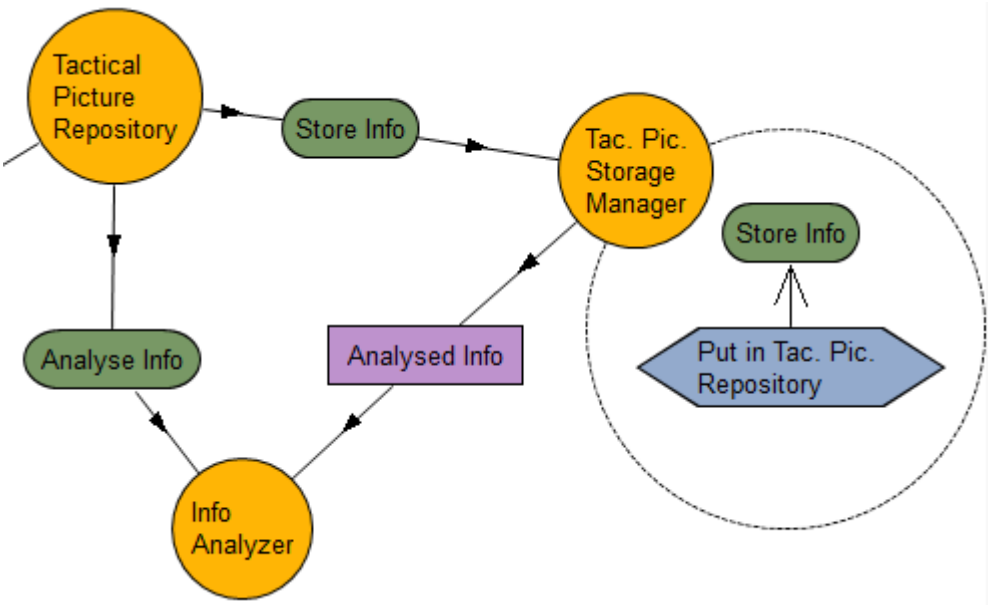


Figure 17. Goal diagram of *Tactical Picture Storage Manager* actor.

Figure 18 gives the combined Goal Diagram of *Tactical Picture Repository* and sub actors.

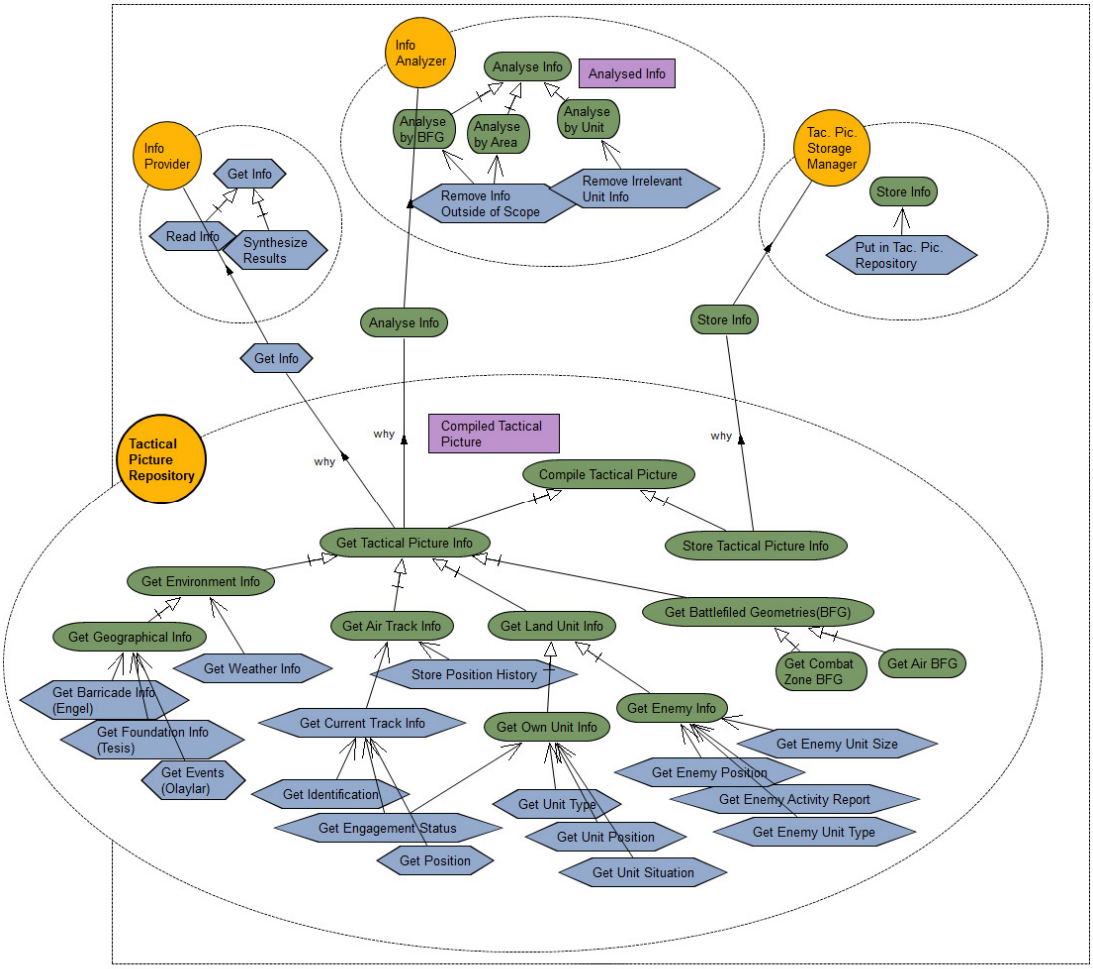


Figure 18. Combined goal diagram of *Tactical Picture Repository* actor and sub actors.

3.4.1.2 Tactical Picture Presenter Actor

Main goals of *Tactical Picture Presenter* actor are showing tactical picture information, providing display functions, providing analysis and filter operations. These goals lead us to define actors for fulfilling each of them. First goal, showing the tactical picture information, is remained as the goal of *Tactical Picture Presenter* actor. Other goals can be assigned to new actors to fulfill them. By separating these features, *Tactical Picture Presenter* actor just presents the tactical picture. Other features about Tactical Picture operations are achieved by other actors. The feature, loading maps and managing map database, can also be separated from *Tactical Picture Presenter*, therefore a new actor is defined for map operations.

We introduce 4 new actors under *Tactical Picture Presenter* actor at this phase: *Filter Manager*, *Map Manager, Display functions Manager* and *Analysis Manager*. We also introduce a new actor, *External Device Manager*, outside of the *Tactical Picture Presenter* actor. Later, *Map Manager* actor will depend on this actor to reach external devices. *Operations UI* actor was previously introduced in the late requirement phase. Actor diagram of *Tactical Picture Repository* is depicted in Figure 19.

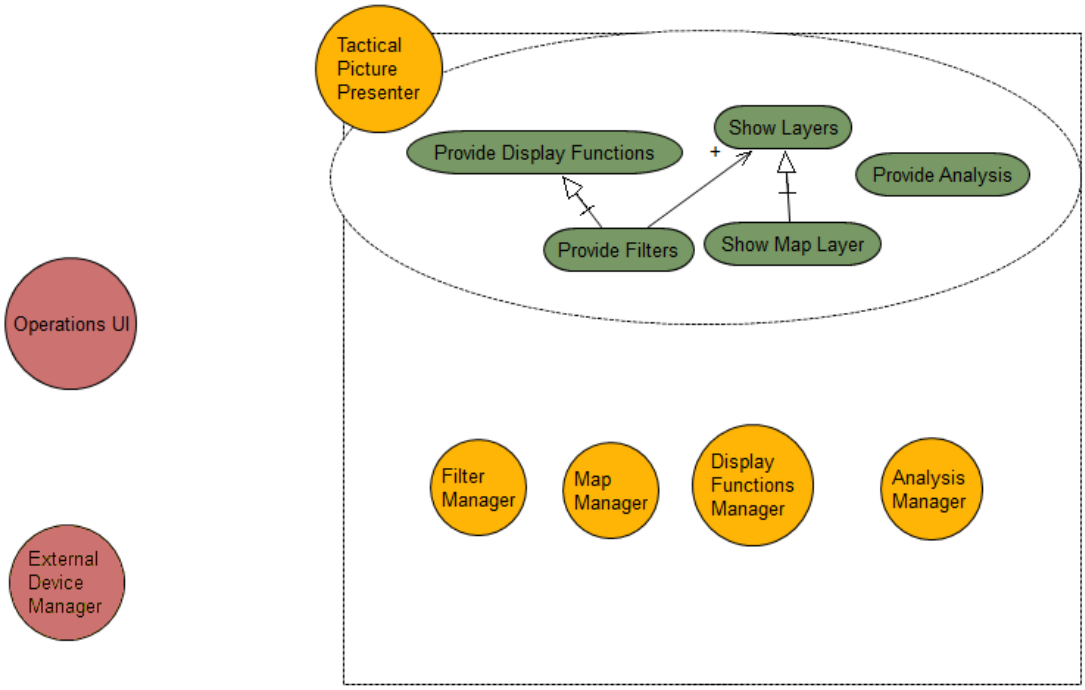


Figure 19. Actor diagram of Tactical Picture Repository actor.

Filter Manager actor expands the *Provide Filters* goal of *Tactical Picture Presenter* actor. Filters enables user to customize the tactical picture. By using filters, user may choose the information to show/hide.

In this work, for the sake of simplicity, we defined filters given in Table 1. In real systems, filters may be more complicated.

Table 1. Filter types**.**

Map layer filter
Vector layer filter
Grid line filter
Raster map filter
Relief map filter
Unit Filter
Show all/hide all
Planned/current state filter
Type based filter
Sensor filter
Weapon filter
Command center filter
…
Real/Simulated unit filter
Additional information
Primary target line filter
Weapon range filter
Sensor radiation sector filter
Force relation filter
…
Track Filter
Show all/hide all filter
Type filter
Fixed Wing
Rotary Wing
…
Identification filter
Unknown
Friendly
Hostile
…
Real/simulated filter
…
BfG Filter
Show all/hide all filter
Source filter (defined or received)
Type filter
Ground BfG filter
Air BfG filter
Events filter
…

Goal Diagram of *Filter Manager* actor is depicted in Figure 20. To keep the model simple, we included only some of filters.

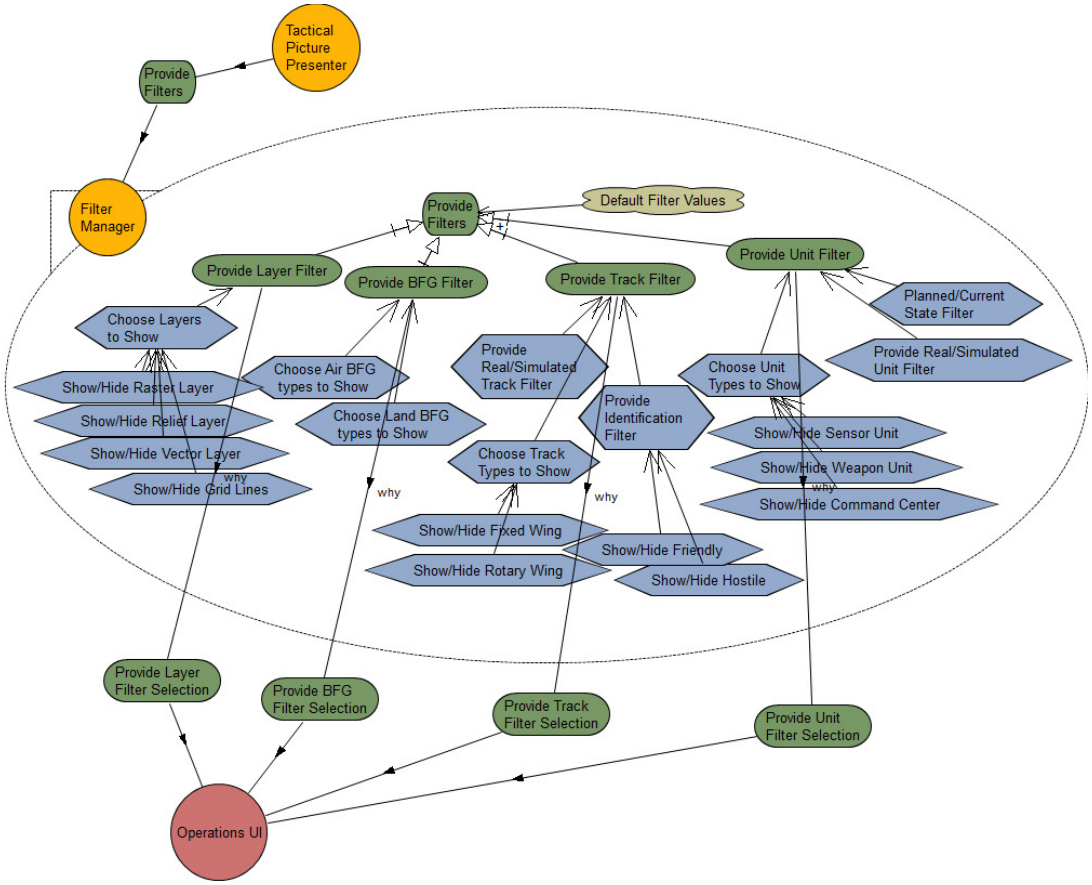


Figure 20. Goal diagram of Filter Manager actor.

Tactical Picture Presenter actor depends on *Filter Manager* actor by the goal *Provide Filters*. User may want to change filter values. Since *Filter Manager* actor do not have user interface features, this activity is assigned another actor outside of Tactical Picture Presenter actor, which is *Operations UI* actor. *Filter Manager* actor depends on *Operations UI* actor by its UI goals; *Provide Layer Filter Selection, Provide BfG Filter Selection, Provide Track Filter Selection* and *Provide Unit Filter Selection*.

Map Manager actor is defined for loading and managing maps. First, it asks user which type of maps will be loaded. *Map Manager* actor uses *Choose Map Type* goal of Operations UI to. Then, it searches external devices and find relevant

files. Finally, it copies map files from external devices such as USB, CD/DVD. This actor also stores and manages loaded maps.

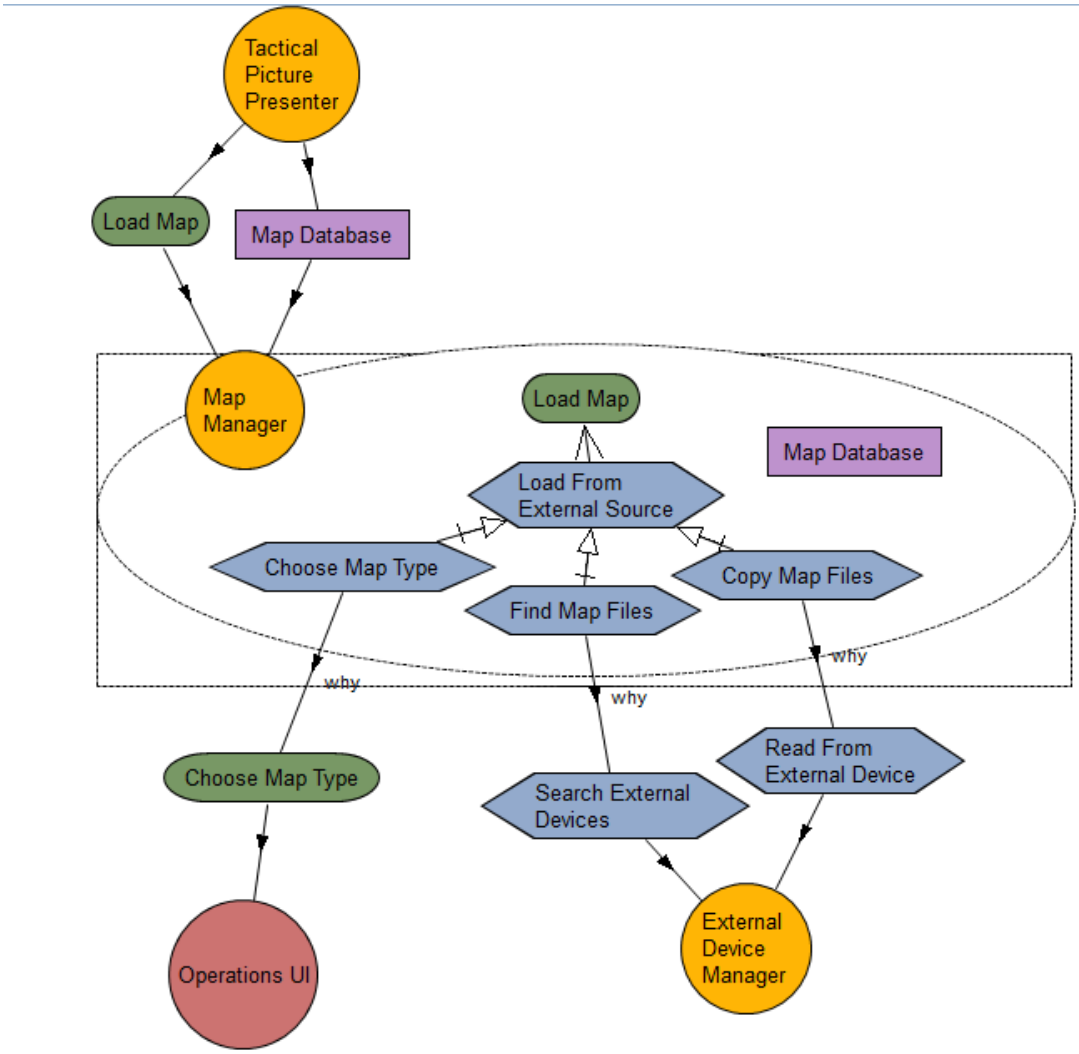


Figure 21. Goal diagram of *Map Manager* actor.

Tactical Picture Presenter actor depends on this actor by using its goal *Load Map*. *Map Manager* actor depends on *External Device Manager* actor by using its plans, *Search External Devices* and *Read From External Device*. Goal Diagram of *Map Manager* actor is depicted in Figure 21 and goal diagram of *External Device Manager* actor is illustrated in Figure 22.

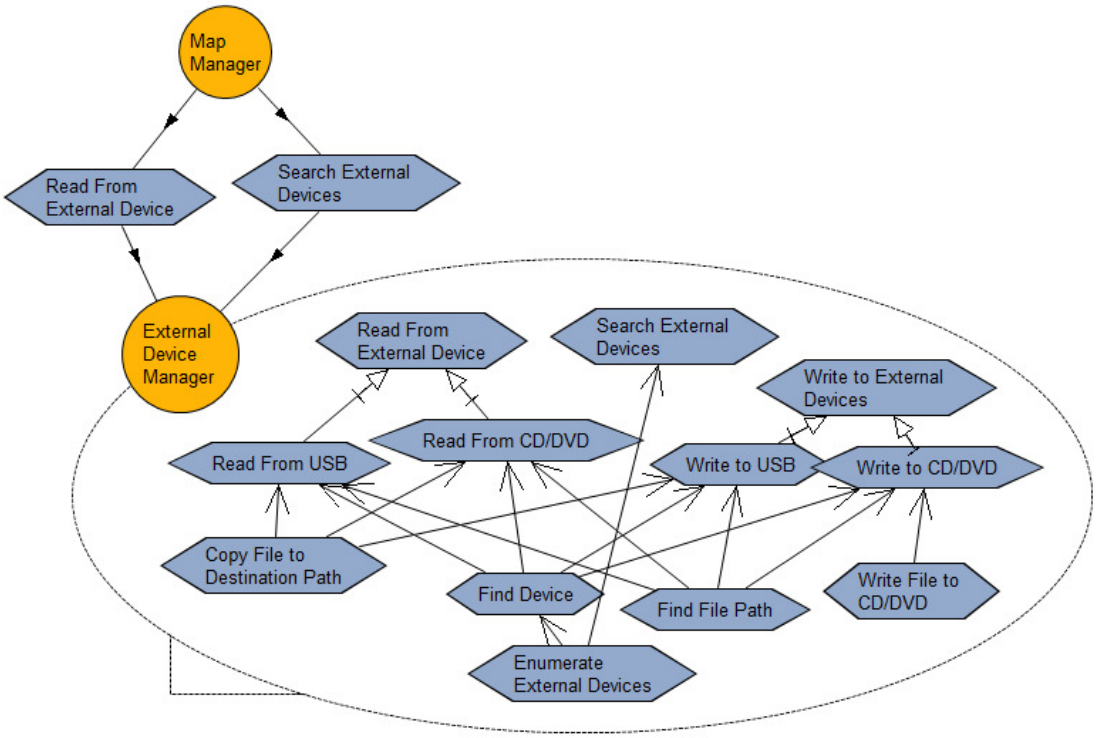


Figure 22. Goal diagram of *External Device Manager* actor.

The *Analysis Manager* actor aims to provide map layer analysis. Analysis types modeled in this thesis are profile analysis, altitude query, visibility analysis, distance analysis and direction angle analysis. Each analysis basically is composed of two steps. The first is to get user data for the analysis. The second is to do the analysis and produce the result. Required user data changes according to the type of analysis. Results also change according to the type of analysis. There may be single value results, or graphical results.

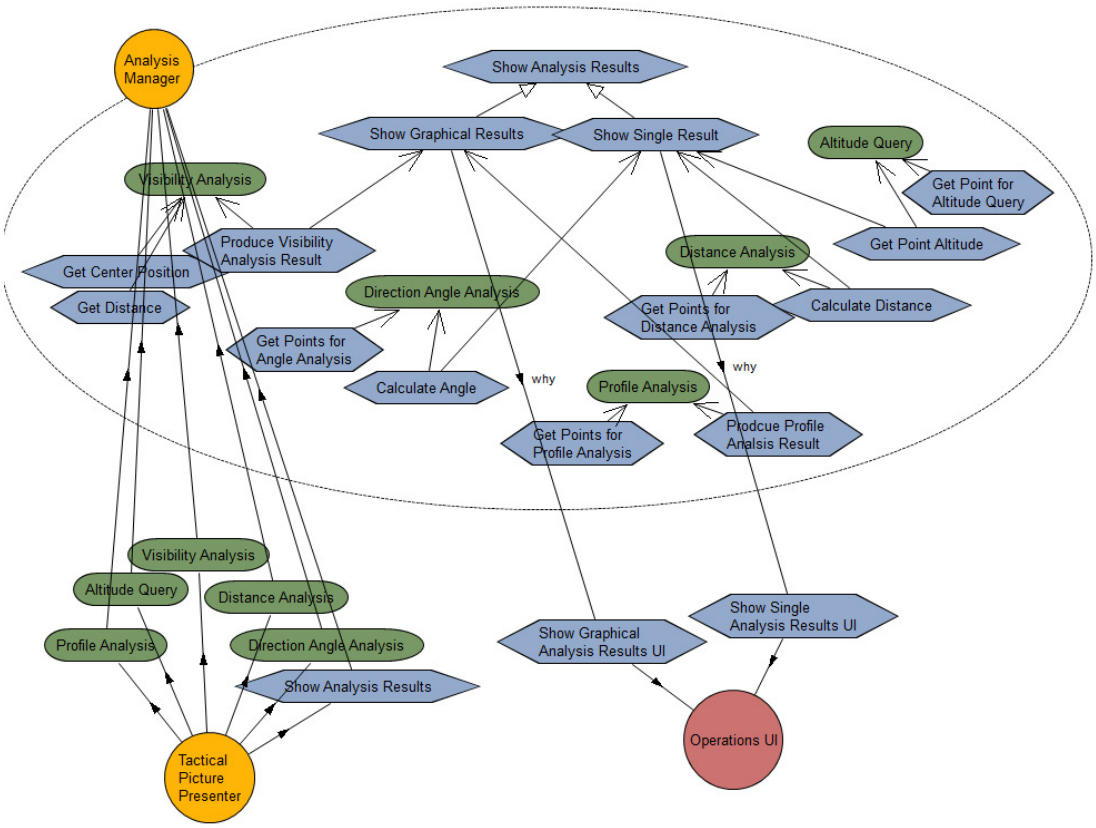


Figure 23. Goal diagram of *Analysis Manager* actor**.**

Analysis Manager actor uses capabilities of *Operations UI* actor to show analysis results. *Tactical Picture Presenter* actor depends on *Analysis Manager* actor by its analysis goals and *Show Analysis Results* plan. Goal Diagram of *Analysis Manager* actor is depicted in Figure 23.

Display Functions Manager actor provides tactical picture operations like symbol selection, zooming and panning operations. In the tactical picture, all symbols should be selectable. User can select a unit symbol, BfG symbol or track symbol. When user looks at tactical picture, s/he should easily recognize the selected symbol. For the easy recognition, selected symbol must be recognizable; therefore selected symbol should be drawn with different color than other symbols. To represent this issue in our model, *Easy Recognition of*

Selected Symbol is defined as soft goal, which is supported by *Draw with Different Color* plan under *Show Selected Symbol Differently* goal.

In the tactical picture, zoom operations are helpful for user to understand the situation. By zooming in and zooming out, user can change the scope and see area at the desired details level. Zoom operations can be done using mouse or keyboard. New zoom level can be decided by given scale. System can be zooming to a predefined value i.e. beginning zoom level, or to a user defined value.

Panning can be defined as moving over tactical picture, i.e. moving to left or up, without changing the zoom level. User may want to pan to selected symbol, which helps to recognize the selected symbol. User may also want to pan to user defined coordinate. Mouse or keyboard can be used for panning operations.

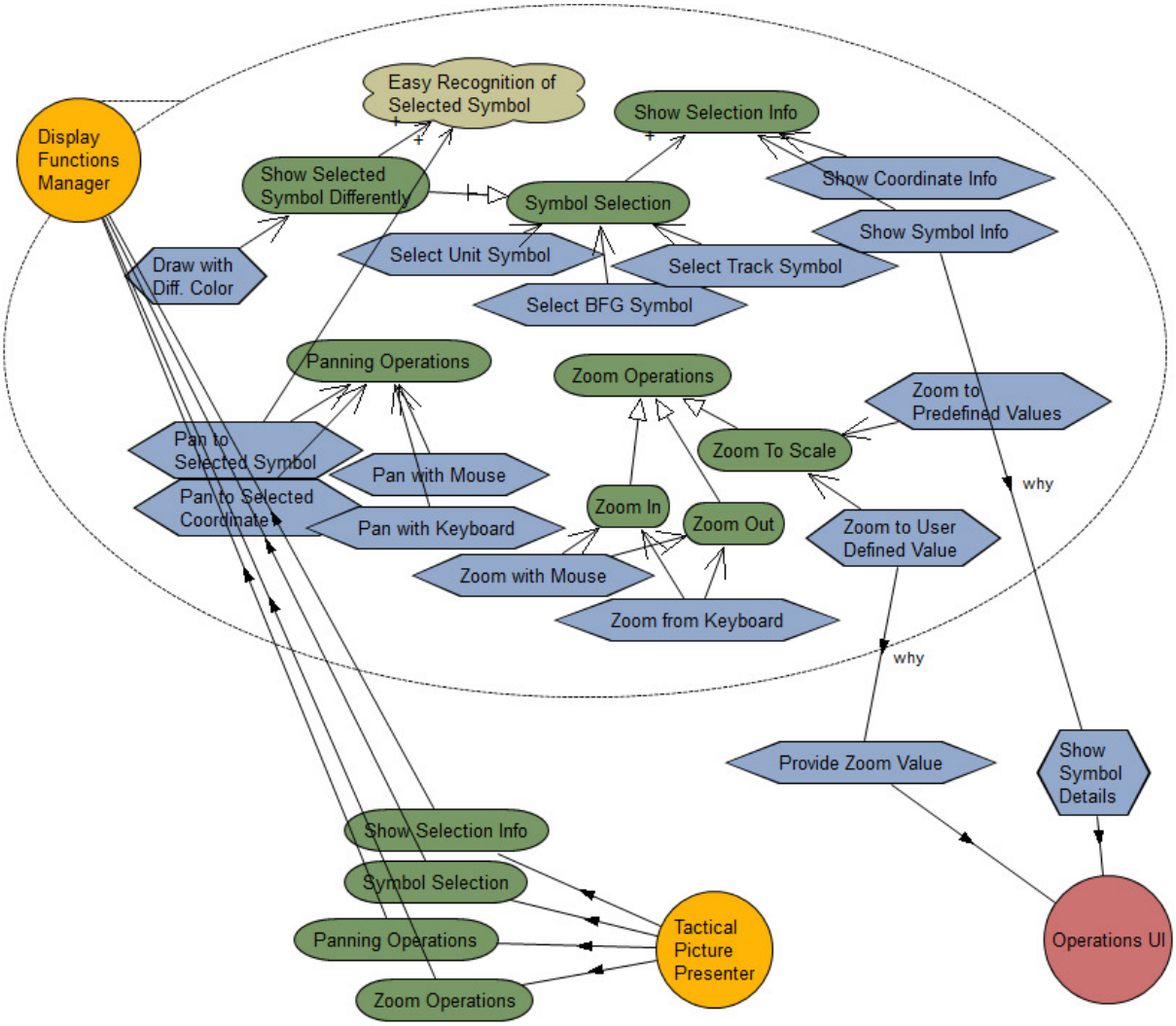


Figure 24. Goal diagram of *Display Functions Manager* actor.

Goal Diagram of *Display Functions Manager* actor is depicted in Figure 24.

Figure 25 presents the Goal Diagram of *Tactical Picture Presenter* at the architectural design phase. Figure 26 illustrates the combined Actor Diagram of *Tactical Picture Presenter* and sub actors and their relations.

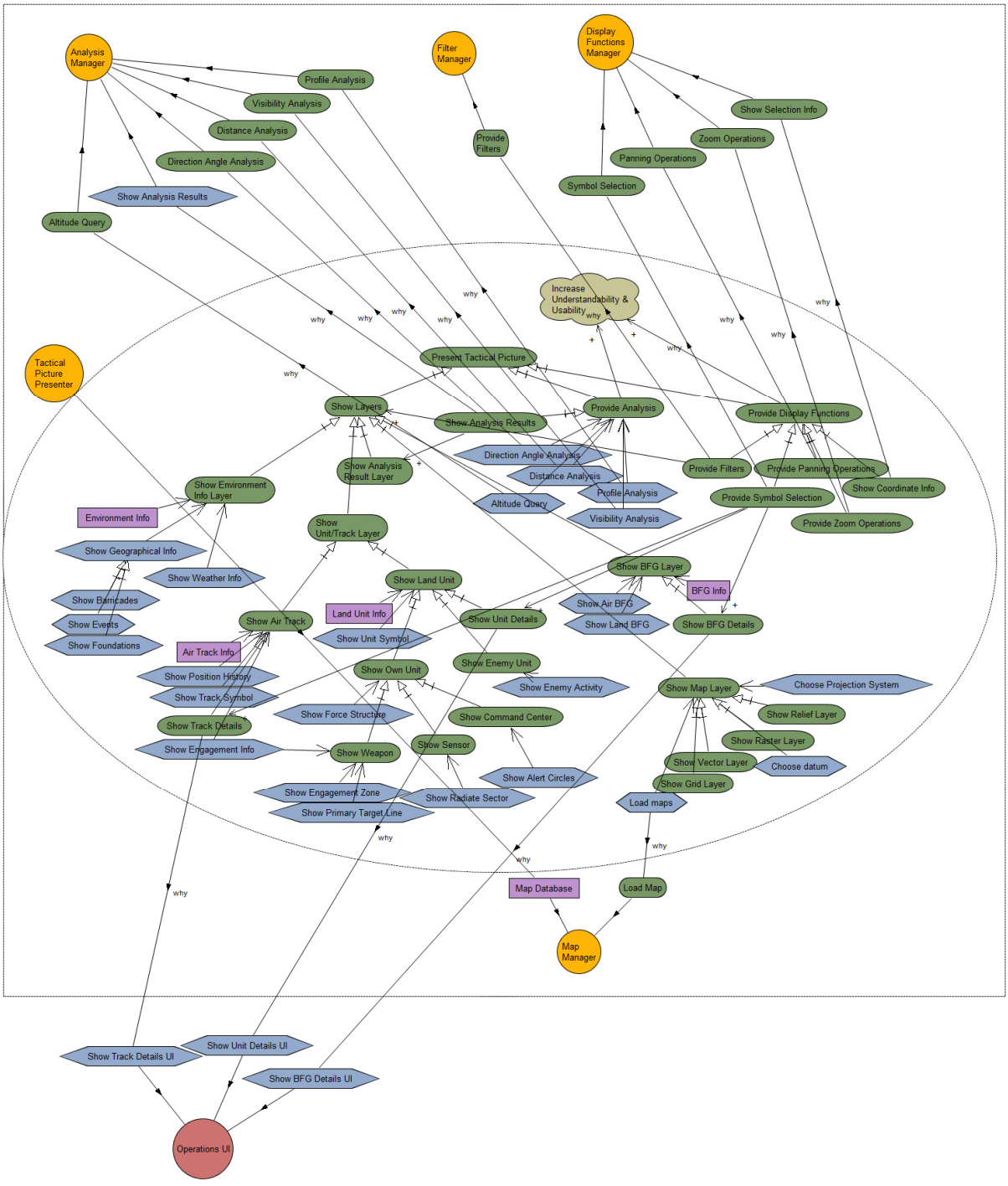


Figure 25. Goal diagram of *Tactical Picture Presenter* actor.

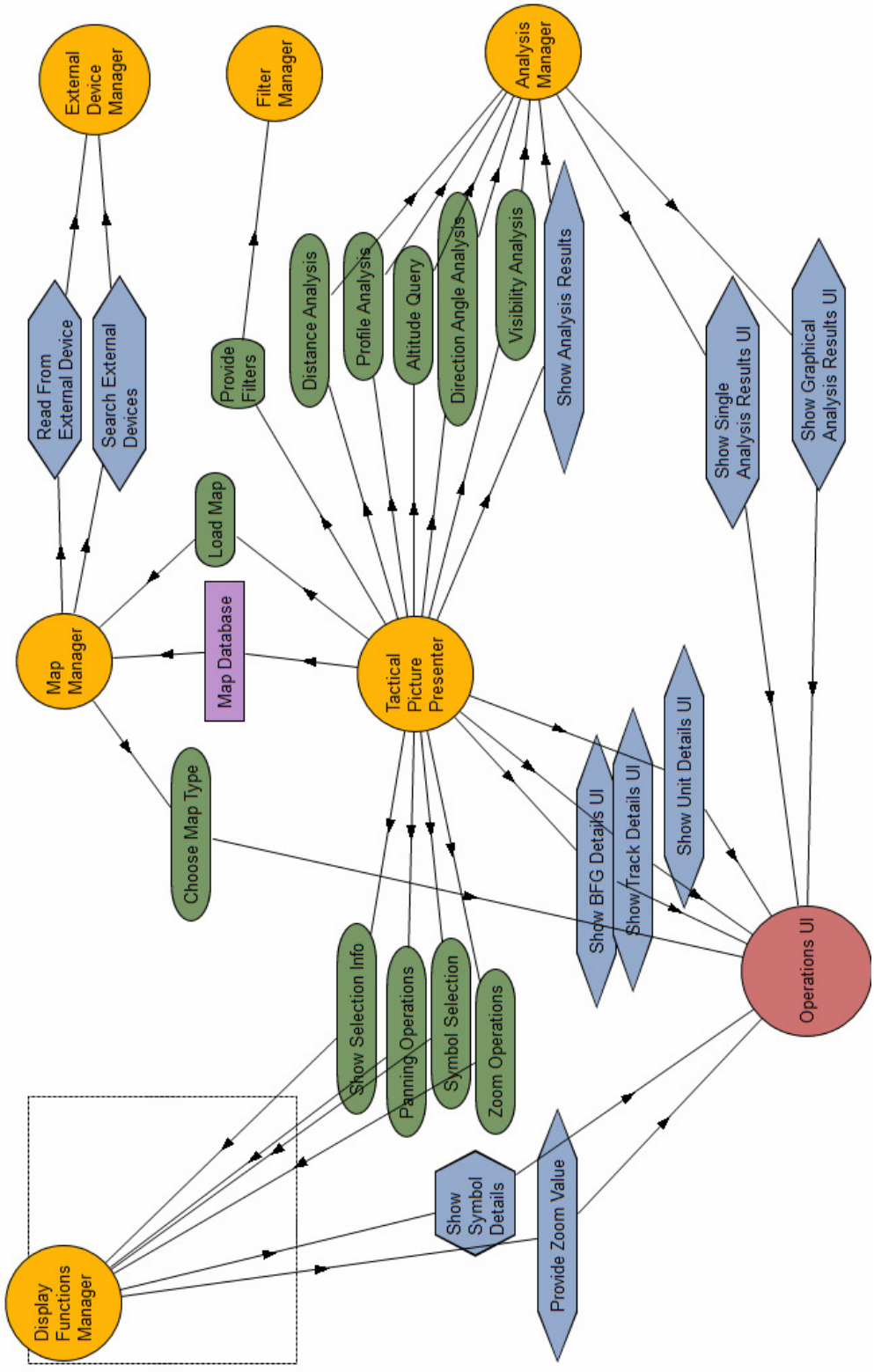


Figure 26. Combined actor diagram of *Tactical Picture Presenter* actor and sub actors.

3.4.2 Step 2: Identifying Capabilities

This step consists in the identification of the capabilities needed by the actors to fulfill their goals and plans. Capabilities are identified by analyzing the extended actor diagrams. A plan with means-end relation to a goal gives a capability. Capabilities are automatically generated by the TAOM4E tool.

Table 2 gives the capabilities of actors.

3.4.3 Step 3: Agents and Capability Assignments

This step consists of defining a set of agent types and assigning each of them different capabilities. Since we analyzed just one part of whole C2 system, we ended with a few agents and capabilities. In the whole architecture, there will be many other capabilities and agent types. For a real system design, complete extended actor diagram must be produced to be able to identify whole agents and capabilities.

Agent assignment depends on the designer. Agents and capability assignments are affected by the analysis of the extend actor diagram. Also, designer's view of the system in term of agents affects the design.

A simple way to define agents is to assign similar jobs to same agents. To do that, we first group capabilities. An agent should provide similar capabilities; therefore a group of capabilities can be assigned to an agent. On the other hand while defining agents, we have to consider system design, i.e. system-to-be will work with these agents.

Table 2. Actors' capabilities.

Actor Name	No	Capability
Analysis Manager	1	Get Points for Distance Analysis
	2	Calculate Distance
	3	Get Points for Angle Analysis
	4	Calculate Angle
	5	Get Center Position
	6	Produce Visibility Analysis Result
	7	Get Point for Altitude Query
	8	Calculate Altitude
	9	Get Points for Profile Analysis
	10	Produce Profile Analysis Result
	11	Get Distance
Display Functions Manager	12	Show Symbol Info
	13	Show Coordinate Info
	14	Select Unit Symbol
	15	Select BfG Symbol
	16	Select Track Symbol
	17	Pan to Selected Coordinate
	18	Pan with Mouse
	19	Pan to Selected Symbol
	20	Draw with Diff. Color
	21	Zoom with Mouse
	22	Zoom from Keyboard
	23	Zoom with Mouse
	24	Zoom from Keyboard
	25	Zoom to User Defined Value
	26	Zoom to Predefined Values
	27	Pan with Keyboard
Filter Manager	28	Choose Air BfG types to Show
	29	Choose Land BfG types to Show
	30	Choose Layers to Show
	31	Choose Unit Types to Show
	32	Choose Track Types to Show
	33	Provide Real/Simulated Unit Filter
	34	Planned/Current State Filter
	35	Provide Real/Simulated Track Filter
	36	Provide Identification Filter
Information Analyzer	37	Remove Irrelevant Unit Info
	38	Remove Info Outside of Scope
	39	Remove Info Outside of Scope
Map Manager	40	Load From External Source
Operations UI	41	Provide Zoom Value
	42	Show Unit Details UI
	43	Show BfG Details UI
	44	Show Track Details UI
	45	Show Single Analysis Results UI
	46	Show Graphical Analysis Results UI

Table 2 (Continued)

Actor Name	No	Capability
Tactical Picture Presenter	47	Direction Angle Analysis
	48	Distance Analysis
	49	Altitude Query
	50	Profile Analysis
	51	Visibility Analysis
	52	Show Position History
	53	Load maps
	54	Choose Projection System
	55	Choose datum
	56	Show Air BfG
	57	Show Land BfG
	58	Show Track Symbol
	59	Show Engagement Info
	60	Show Engagement Info
	61	Show Primary Target Line
	62	Show Engagement Zone
	63	Show Radiate Sector
	64	Show Unit Symbol
	65	Show Force Structure
	66	Show Alert Circles
	67	Show Enemy Activity
	68	Show Weather Info
	69	Show Geographical Info
	85	Show Raster Layer
	86	Show Relief Layer
	87	Show Vector Layer
	88	Show Grid Layer
Tactical Picture Repository	70	Get Enemy Unit Size
	71	Get Enemy Position
	72	Get Enemy Unit Type
	73	Get Enemy Activity Report
	74	Get Unit Position
	75	Get Unit Type
	76	Get Unit Situation
	77	Get Engagement Status
	78	Get Weather Info
	79	Get Barricade Info
	80	Get Foundation Info
	81	Get Events
	82	Store Position History
	83	Get Current Track Info
Tactical Picture Storage Manager	84	Put in Tactical Picture Repository

There are user interface capabilities. We divide user interface into two parts; Tactical Picture UI and Operations UI. Tactical Picture UI directly operates on Tactical Picture, i.e. drawing symbols, showing map layers, zoom operations, selecting symbols etc. On the other hand, Operations UI provides user additional information and gets user input i.e. showing symbol details, getting filter choices, showing analysis results etc. We define two agents for these UI parts as Tactical Picture UI Agent and Operations UI Agent. Then we assign each agent relevant capabilities. Agents and assigned capabilities are given in Table 3.

Table 3. Agent types and their capabilities.

Agent	**Capabilities**
Tactical Picture UI Agent	1, 3, 5, 7, 9, 11, 14-27, 52, 56-69
Storage Agent	82, 84
Storage Broker Agent	37-39, 70-81, 83
Filter Agent	33-36
Analysis Agent	2, 4, 6, 8, 10, 47-51
Map Manager Agent	40, 53
Operations UI Agent	12, 13, 28-32, 41-46, 54, 55

We add two agents for storage capabilities. First agent, *Storage Broker Agent*, gets the information from the Data Repository. Another capability of this agent is to analyzing information i.e. remove unnecessary information. Other agent, *Storage Agent*, stores the tactical picture information and presents to Tactical Picture UI.

Our SA system has filter capabilities. These capabilities lead us to add *Filter Agent.* This agent is responsible for filter operations. *Filter Agent* keeps user filter choices and informs *Tactical Picture UI Agent* which information to show/hide.

Analysis capabilities are assigned to *Analysis Agent*. This agent gets required input from UI agents; produces results and then presents to UI agents to show to user. Finally an agent is defined for managing maps as *Map Manager Agent*.

CHAPTER 4

DISCUSSION

4.1 Achievements

This work presented the goal oriented modeling of the SA component in a C2 Air Defense System. Using goal oriented modeling enabled us to easily find and justify the requirements of a system. Notion of goals provided us to identify, organize and justify system requirements. Notion of actors enabled us to find and model the stakeholders. Actor and goal relations provided to easily find and justify the requirements of a system by analyzing the relations.

By using goal oriented approach, we have captured *why* the system is developed. For example, *Track Manager* actor at the early requirement analysis phase, exists because it is responsible for providing track information to *Tactical Picture Provider* actor. *Sensor* actor just provide track data, therefore there is a need for actor to produce track information from track data. Its requirements defines that it does track management, identification etc. but the reason why this actor is developed, can be understood by analyzing the actor relations.

Goal oriented modeling enabled us to investigate system's both functional and non-functional requirements. For example, at early requirements phase, *Commander* actor has *Make Decision* goal. This functional requirement must be analyzed with the quality of decision which is a non-functional requirement. We modeled this non-functional requirement as a soft goal as *Increase Quality of Decision*. If the *Commander* actor would be analyzed at a late requirement analysis phase, we will need to model the actor in a way that this soft goal is satisfied. As another example;

Tactical Picture Provider actor has *Increase Quality of Tactical Picture* soft goal. At the late requirement analysis phase, *Tactical Picture Presenter* actor has *Increase Understandability & Usability* soft goal to achieve previous soft goal. Other actors at the late requirement analysis phase, if analyzed in detail, would have additional soft goals to achieve *Increase Quality of Tactical Picture* soft goal.

Using goal oriented modeling enabled us to understand the environment where the system must operate, and to understand the interactions that should occur between system and environment. At the early requirement analysis phase, *Environment* actor is an example for this relation. *Operations UI* and *Data Repository Manager* actor are defined at the late requirement analysis phase, and no analyzed at the architectural design phase. As we focused on *Tactical Picture Presenter* and *Tactical Picture Repository* actors at the architectural design phase, *Operations UI* and *Data Repository Manager* actors can be viewed as external actors. Since sub actors of *Tactical Picture Presenter* and *Tactical Picture Repository* actors must operate in this environment, we model interaction between these actors as the relations between system and environment.

4.2 Limitations of Our Model

We modeled the SA in a C2 system with applying early requirements, late requirements and architectural design phases of Tropos methodology. At the early requirement analysis phase, model includes all entities of C2 system. By focusing the requirements of Air Defense system, C2 actors are modeled at the knowledge level. 6 actors and 31 goals are defined at this phase. No plans are defined at this phase. We focused on 1 actor, *Tactical Picture Provider*, for the late requirement analysis phase and defined 7 sub actors. 2 (*Tactical Picture Repository* and *Tactical Picture Presenter*) of 7 actors were related to SA; therefore, these 2 actors are analyzed at the late requirement analysis phase and produced 43 goals and 45 plans. At the architectural design phase, *Tactical Picture Repository* and *Tactical Picture Presenter* actors from late requirement analysis phase are analyzed. As a result of

analysis, we generated 9 actors with 56 goals and 57 plans. Totally our model has around 250 entities. Table 4 presents the model statistics at each level.

We tried to keep model simple that one can understand the general concept. We did not go into the details of some parts. For example, we mentioned some important information of tracks, such as position, identification and engagement status. In the model, we presented track information as a resource that combines all information. We presented only engagement status information, because it is related to subordinate. We decided that this level of modeling is enough to illustrate the concept. Track information would be analyzed in much more details at the architectural design phase, but it would just increase the model complexity. In the same way, we did not analyzed BfG types in details. In real systems, a complete model would contain hundreds of actors and thousands of goals, plans, and resources.

Table 4. Model statistics.

Phase	# of Actor	# of Goal	# of Plan	# of Resource
Early Requirement	6	31	0	5
Late Requirement	7	43	45	5
Architectural Design	9	56	57	7
Total	22	130	102	17

4.3 Limitations of Tropos

Tropos methodology was complete and provided detailed descriptions. Also there was a good tool support, TAOM4E, for Tropos modeling. Tropos methodology and tool were sufficient in modeling complex systems. However, we have some suggestions to improve the methodology and tool.

First, we propose some additions to Tropos methodology. Goals/plans that are decomposed under same root goal/plan may have been executed in order. There is

no way of directly showing order of execution. In some cases, dependency relations reveal the order. For example, *Compile Tactical Picture* and *Present Tactical Picture* goals under *Tactical Picture Provider* actor, must be executed in mentioned order. The execution order is obvious, because *Compile Tactical Picture* goal contributes to *Present Tactical Picture* goal. Contributing goal must be achieved before contributed goal. But in other cases, there is a need to show sequential entities in the model. For example, sub goals of *Be Aware of Situation* goal under *Commander* actor at the early requirement analysis phase, depicted in Figure 9, are sequential goals that must be achieved in order as *Perceives Elements of Current Situation, Comprehend Current Situation,* and *Project to Future Status*. At later phases, detailed design may reveal the execution sequence by adding new sub goals and dependency relations. However, we think, it is necessary to show the order of execution at the each level. In our model, we showed the sequence by placing entities from left to right in execution order.

External actors, like *Environment* actor in our model, should be distinguished from system actors. This differentiation helps to separate external interface of the system from the internal interface.

Resources in methodology are defined as used sources, not as generated sources. In technical term, any entity cannot contribute to a resource. We think they should. For example, in Figure 9, *Detect Targets* goal under *Sensor* actor must contribute to *Track Data* resource, but it is not allowed by the tool. Track data is generated information as a result of detect targets action. *Detect Targets* goal must create and maintain the *Track Data* resource. This relation is not shown in our model, because tool does not allow that.

There is a need for resources to be decomposed with AND/OR relations. For example, resources at the late requirement analysis phase, see Figure 13, should be sub resources of *Compiled Tactical Picture* resource.

4.4 Limitations of the Tool

According to methodology, many actors may have dependency relation with the same entity of an actor, but tool does not allow. Methodology also supports positive or negative contributions, but tool supports only positive one.

Tool support for Tropos is currently only in the form of a diagram editor. There is a need to include more rules to tool. For example, there is a need for the consistency checking of the model. Another issue is that, tool support for transition between phases is weak. For example, tool does not have capability to show the relation between a goal at the early requirement analysis phase and generated goals at the late requirement phase. The relation of model entities between phases should be entered by user and showed by the tool. These automatic controls lead user to make fewer mistakes. In complex models, a defined entity at a phase may have been forgotten at later phases. Tool support for transition between phases prevents these kinds of mistakes.

Designer may need adding explanations when modeling with the tool. Tool may have additional capabilities like adding descriptions to model entities actor, plans, and goals. Adding model explanations may also be helpful.

Only capability list can be generated from the tool. There would be more automatic report generation capabilities. For example, tool may automatically generate a design description that includes descriptions for each design entity, or generate a report that shows how a goal at early phases is achieved at the later phases.

4.5 Related Work

Modeling Command and Control is an attractive research area. There are several works [14, 15, 19, 27, 28, 29, 30 and 34] that focus on different types of C2 system, or some part of it. Since C2 is a very broad area, researches focus on some sub parts

of C2 systems i.e. ballistic missile defense system [28]. Some of them model C2 systems at a very abstract level, i.e. modeling C2 in multi-agent systems at the conceptual level [15]. Some models define the system at the mission level, i.e. information management in battle field visualization [29].

Agent oriented methodologies are usually used in the simulation of C2 entities. C2 entities are modeled as agents, and agent interactions are analyzed. These researches [27, 34] try to model C2 entities in a distributed environment.

CABLE [27] is presented as a multi-agent architecture to support C2. It provides a generic framework for building distributed agent based systems together. CABLE models C2 entities as distributed agents. SARA [24] provides task assistance for search and rescue operations. SARA, just like CABLE, models C2 systems as multi-agent collaborative systems.

Taylor [29] describes architecture for information management using agents to assist a commander with battlefield visualization. Information management is presented as the center process of SA. Taylor identifies the early requirements of an agent based system for enabling battlefield visualization and defines the system architecture at an abstract level.

There are several researches that work on air defense systems [19, 29], tactical picture [30] and SA [11, 14 and 19]. But, as far as we know, there is no work for agent/goal oriented modeling/requirement analysis of SA in air defense systems.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This work presented the goal oriented modeling of the SA component in a military C2 System. Air Defense system was chosen as the example system.

The main focus of this work was on requirement analysis of C2 systems. Goal oriented requirement analysis was chosen as analysis the methodology. We narrowed down the scope to SA in C2, to be able to go into details of requirements. Tropos, an agent/goal oriented methodology, is used for analysis and modeling. First 3 modeling activities of Tropos over 5 phases are performed; Early Requirements, Late Requirements and Architectural Design.

At the early requirements phase, we studied the environment for the system-to-be. The relevant stakeholders are identified and interviewed. At this phase, whole C2 system is modeled. Main actors of C2 and their relations are defined. Goal oriented approach helped us to better understand the system-to-be and its environment. Also, goals provided rationale for requirements.

One of the main benefits of goal-oriented requirements analysis was additional support for the early requirements analysis. Goal-oriented requirements analysis approach takes a wider system engineering perspective compared to the traditional requirements analysis methods. Domain properties about the environment are explicitly captured during the requirements elaboration process.

With the help of goal oriented requirement analysis, we modeled the early requirements of C2 according to the C2 conceptual model given in [4]. This single

model showed the main stakeholders of the C2 system and their responsibilities and relations with each other. Produced model explicitly presented the Command and Control process. Goals provided a precise criterion for sufficient completeness of a requirement.

At the late requirement analysis phase, we focused on SA component of C2 system. New actors are defined and modeled at this phase. Goal oriented modeling helped us to define new actors and relations. Goal oriented approach provided traceability between high-level strategic objectives to low-level technical requirements. One can easily follow how a goal at the early requirement analysis is detailed at the late requirement analysis, and how it is achieved at the architectural design phase.

In goal oriented model, actor and goal models provide a natural mechanism for structuring complex requirements. It is more understandable when showing requirements visually instead of just writing them textually. These "visual" models also give the structure of requirements. Goal models can also be used to provide the basis for the detection and management of conflicts among requirements.

At the architectural design phase, we defined the system's global architecture in terms of sub-systems, interconnected through data and control flows. We represented sub-systems as actors and data/control interconnections as dependencies. Capabilities needed by the actors to fulfill their goals and plans, are identified. Then, agents are defined and each capability is assigned to a relevant agent.

Goal oriented modeling approach helped us to easily define the sub-actors and control flows. In our example's architectural design model, some actors can be thought as software/hardware components. Goal oriented modeling approach was successful at modeling inside of these components and interrelation of these components.

Steps 2 and 3 of architectural design phase can viewed as a preparation for agent oriented implementation. Performing the first phase, we think, is enough for the cases that will not continue with agent oriented implementation.

We preferred to use Tropos, which was the most appropriate methodology with its support for early requirement analysis phase and with its modeling tool. After this work, we concluded that methodology and tool, in general, was successful. However, methodology needs some additions and corrections given in Evaluation part. TAOM4E tool also has some restrictions. We could not present some parts of the model as we intended. These are also discussed in Evaluation section.

Tropos methodology can be used up to some phase, after that development may continue with other software development techniques. Tropos methodology can be used with non-agent (e.g., object-oriented) methodologies. For example, Tropos methodology may be used for early development phases and then UML may be used for later phases. Transition from Tropos methodology to other methodologies (object-oriented, feature models, UML, etc.) can be studied as a future work.

Finally, we conclude that that goal oriented methodologies, namely Tropos, can be used for the analysis of software intensive systems for the requirement analysis or further phases, even if the implementation will not be agent oriented. The last phase that will be applied depends of the type of software and designer's choice.

For the future work, whole C2 system may be analyzed up to late requirement analysis phase. This enables to explicitly view the detailed requirements of a C2 system. This could bring to light the power of goal oriented requirement analysis methodologies. Another C2 system, other that Air Defense system, can be modeled as an example to approve the applicability of the model to C2 domain.

As a future work, some additions and changes can be applied to methodology and tool, for C2 domain. Extended methodology may reveal C2 domain specific properties.

REFERENCES

[1] M. Wooldridge, and N. R. Jennings, "Agent Theories, Architectures and Languages: A Survey", *Workshop on Agent Theories, Architecture and Languages*, pp. 1-32, 1995.

[2] DoD Dictionary of Military and Associated Terms. http://www.dtic.mil/doctrine/dod_dictionary, last accessed 20 December 2010.

[3] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), pp. 285-312, 2000.

[4] NATO RTO Technical Report, "Exploring New Command and Control Concepts and Capabilities", TR-SAS-050, 2007.

[5] P. Bresciani, P. Giorgini, F. Guinchiglia, and A. Perini, "TROPOS: An Agent-Oriented Software Development Methodology", *Journal of Autonomous Agents and Multi-Agent Systems*, 8(1), pp. 203-236, 2004.

[6] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos, "High Variability Design for Software Agents: Extending Tropos", *ACM Trans. Autonom. Adapt. Syst.*, 2(4), 2007.

[7] P.-M. Ricordel, and Y. Demazeau, "From Analysis to Deployment: A Multi-agent Platform Survey", *Engineering Societies in the Agents World*, pp. 93-105, 2000.

[8] D. S. Alberts, and Hayes, R. E., "Understanding Command and Control.", CCRP, 2006.

[9] A. Lamsweerde, "Goal-oriented requirements engineering: A guided tour." *In Proceedings of RE'01 - International Joint Conference on Requirements Engineering*, pp. 249–263, 2001.

[10] Yu, E., "Modeling Strategic Relationships for Process Reengineering", *PhD thesis*, University of Toronto, Department of Computer Science, 1995.

[11] M. R. Endsley, "Towards a theory of situation awareness in dynamic environments." *Human Factors*, 37, pp. 32–64, 1995.

[12] J. Odell, H. Parunak, and B. Bauer, ‘‘Extending UML for agents’’, *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pp. 3–17, 2000.

[13] E. Yu, and J. Mylopoulos, "Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering", *International Journal of Intelligent Systems in Accounting, Finance and Management*, 5(1), 1996.

[14] S. Munk, “Situational awareness (data) bases in military command and control”, *Journal of Academic and Applied Research in Military Science*, 3(3), pp. 373–385, 2004.

[15] T. R. Ioerger, Linli He, “Modeling command and control in multi-agent systems”, *8th International Command and Control Research and Technology Symposium*, Washington, DC, 2003.

[16] N. Jennings, K. Sycara, and M. Wooldridge, “A Roadmap of Agent Research and Development” *International Journal of Autonomous Agents and Multi-Agent Systems*, 1(1), pp. 7-38, 1998.

[17] J. Castro, M. Kolp, and J. Mylopoulos, ‘‘Towards requirements-driven information systems engineering: The tropos project,’’ *Information Systems*, vol. 27, pp. 365-389, 2002.

[18] A. van Lamsweerde, and E. Letier, “From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering”, *Proc. Radical Innovations of Software and Systems Engineering*, LNCS, 2003.

[19] M. G. Oxenham, “Enhancing Situation Awareness for Air Defence via Automated Threat Analysis”, *Proceedings of the Sixth International Conference of Information Fusion*, 2003.

[20] B. Bauer, J. P. Müller, and J. Odell, “Agent UML: A formalism for specifying multiagent software systems”, *Int. J. Software. Eng. Knowl. Eng.*, vol. 11, no. 3, pp. 207–230, 2001.

[21] L. Padgham, and M. Winikoff, “Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents”, *In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, pp. 97–108, 2002.

[22] S. A. DeLoach, “Multiagent Systems Engineering A Methodology and Language for Designing Agent Systems”, *In Proc. of Agent Oriented Information Systems*, pp. 45–57, 1999.

[23] A. Lapouchnian, "Goal-Oriented Requirements Engineering: An Overview of the Current Research", *Depth Report*, University of Toronto, 2005.

[24] K. S. Nelson, R. R. Penner, and N. H. Soken, "SARA: A Multi-Agent System for Collaborative Command and Control", *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, pp. 784 – 793, 1998.

[25] M. Coburn, "Jack intelligent agents: Agent Manual 5.3", 2008, http://www.agent-software.com, last accessed 20 December 2010.

[26] M. Bratman, "Intention, Plans, and Practical Reason", Harvard University Press, Cambridge, MA, 1987.

[27] C. Dee, P. Millington, B. Walls, and T. Ward, "CABLE: A multi-agent architecture to support military command and control", *Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)*, 1998.

[28] D. Wijesekera, J. B. Michael, A. Nerode, "BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies", *In Tenth Command and Control Symposium*, McLean, VA, 2005.

[29] G. Taylor, S. Wood, K. Knudsen, "Enabling Battlefield Visualization: An Agent-based Information Management Approach", *10th International Command And Control Research And Technology Symposium*, 2005.

[30] E. P. O'Shaughnessy, S. P. Bradley, D. McGeechan, "Automating the Force Tactical Picture", *Command and Control Research and Technology Symposium*, 2000.