A Domain Modeling Language for Military Command, Control and Information Systems

Harald Rønneberg

Information Systems Group
Faculty of Physics, Informatics and Mathematics
The Norwegian University of Science and Technology

15 January 1997
Abstract

It is widely recognized that deep domain knowledge as well as good communication between the users and the information system (IS) engineers in the requirements engineering stage are vital in ensuring that an information system will be in accordance with the users' needs, and developed within budget and time schedule. Contemporary information system models are used to enhance communication and understanding between the users and the IS engineers. However, studies show that contemporary modeling languages have not removed all communication problems between users and IS engineers. Studies also show that IS engineers too often have insufficient domain knowledge and that IS engineers are in power through their IS models.

We propose to improve on this imbalance by introducing domain languages, ie languages that are comprehensible to both users and IS engineers, are able to express how the domain works, enable IS engineers and users to have sufficient understanding of the domain, and dramatically reduce the communication problems between IS engineers and users. To be able to meet these requirements a domain language has to be: based on the users' professional language, developed on the behalf of the users (ie domain experts), and closer related to traditional IS modeling languages than natural languages are.

We have investigated the domain of military command, control and information systems. On the basis of the most important characteristics of command, control and information system we have suggested a multiple perspective domain language. With aid of the domain language it is possible to view the domain in six different perspectives: process, goal, role, organization, information, and IT support.

We apply six sub-languages and four information tables in order to visualize these six perspectives. The six sub-languages are: the process overview sub-language, the process sub-language, the requirement, goal, problem and mean sub-language (RGPM sub-language), the message sequence sub-language, the role sub-language, and the organization structure sub-language. The four information tables are: the message content table, the responsibility table, the rule table, and the goal table. Note that we have no separate IT modeling sub-language. The IT modeling sub-language is part of the process sub-language and the message sequence sub-language.

The developed domain language is firmly based on practical usefulness and it is based on a domain formalism that is a standard among the military.
Preface

The work presented in this report is a thesis submitted to The Norwegian University of Science and Technology (NTNU) for the doctor degree "doktor ingeniør". The work has been carried out at the Information Systems Group, Faculty of Physics, Informatics and Mathematics, NTNU, Trondheim, Norway under the supervision of Professor Dr. Arne Sølvberg.

Acknowledgments

I would like to thank the following persons who have supported me during the work with the thesis:

• Professor Arne Sølvberg, my supervisor, for his continuous encouragement, guidance, and constructive criticism throughout the work.
• Colonel Erik Hammer for making it possible for me to start the Ph.D.-studies already when I worked for the Norwegian Army Materiel Command.
• The members of the artillery command, control and information system project ODIN 2, especially lieutenant-colonel Kjell Kringsjå, captain Arne Knapskog, captain Rune Kvarnstrøm, lieutenant Arnstein Ødegård, lieutenant Ola Wiker, researcher Camilla Bårnes and researcher Geir Kåre Sletten who have been invaluable discussion partners. Without their support and artillery domain knowledge I would not have been able to develop a domain modeling language for command, control and information systems.
• Fellow students at the Information Systems Group: Terje Brasethvik, Gunnar Brataas, Babak Amin Farshchian and Tom Reidar Henriksen for providing an inspiring working environment.
• Ståle A. Olsen for building a prototype drawing editor of the domain language suggested in the thesis.
• The students I have supervised as part of my Ph.D.-studies, especially Terje Brasethvik, Kim Foshaug and Ståle A. Olsen who did literature studies on contemporary modeling languages and other user oriented development approaches.
• My wife Beate and my son Jørgen for supporting me all the way through.

Trondheim, January 1997

Harald Rønneberg
iv
Contents

1 INTRODUCTION .................................................................................................................. 1
  1.1 PROBLEM DESCRIPTION ........................................................................................... 1
  1.2 MOTIVATION ............................................................................................................. 4
  1.3 GOAL ............................................................................................................................ 5
  1.4 APPROACH ................................................................................................................ 5
  1.5 MAJOR CONTRIBUTIONS .......................................................................................... 5
  1.6 OUTLINE OF THE THESIS ...................................................................................... 6

2 COMMAND, CONTROL AND INFORMATION SYSTEMS ...................................................... 7
  2.1 INTRODUCTION .......................................................................................................... 7
  2.2 CHARACTERISTICS OF C&IS ................................................................................. 10
  2.3 CONTEMPORARY DEVELOPMENT APPROACH FOR C&IS ................................. 12

3 MILITARY SYMBOLS FOR LAND BASED SYSTEMS ....................................................... 15
  3.1 INTRODUCTION TO MILITARY SYMBOLS FOR LAND BASED SYSTEMS ............. 15
    3.1.1 Color presentation .................................................................................................. 15
    3.1.2 Basic symbols ....................................................................................................... 16
    3.1.3 Equipment symbols ............................................................................................. 19
    3.1.4 Fields .................................................................................................................... 19
    3.1.5 Examples showing composite use of some fields .............................................. 25
  3.2 MILITARY SYMBOLS USED IN OUR DOMAIN LANGUAGE ..................................... 27
    3.2.1 Basic symbols ....................................................................................................... 27
    3.2.2 Equipment symbols ............................................................................................. 28
    3.2.3 Fields .................................................................................................................... 28

4 DOMAIN REDESIGN AND REQUIREMENTS ENGINEERING ........................................... 29
  4.1 INTRODUCTION .......................................................................................................... 29
  4.2 REQUIREMENTS ENGINEERING .............................................................................. 31
  4.3 DOMAIN REDESIGN .................................................................................................. 33

5 LANGUAGE REQUIREMENTS .......................................................................................... 37
  5.1 LANGUAGE QUALITY ................................................................................................. 37
    5.1.1 Domain appropriateness ...................................................................................... 38
    5.1.2 Participant knowledge appropriateness ............................................................. 39
    5.1.3 Participant interpretation enhancement ............................................................ 40
    5.1.4 Technical actor interpretation enhancement .................................................... 40
  5.2 DOMAIN LANGUAGE REQUIREMENTS .................................................................... 40

6 MODELING LANGUAGES AND APPROACHES ................................................................ 43
  6.1 DOMAIN DEPENDENT MODELING LANGUAGES .................................................... 43
    6.1.1 The Florence-project ........................................................................................... 43
    6.1.2 The St. Thomas' Hospital project ......................................................................... 44
    6.1.3 The C2RA - NORCCIS II methodology ........................................................... 45
    6.1.4 The GÖTA method .............................................................................................. 47
6.1.3 SAMPO
6.1.6 Object oriented modeling languages
6.2 DOMAIN INDEPENDENT MODELING LANGUAGES
   6.2.1 PPP
   6.2.2 OMT
   6.2.3 OOram
   6.2.4 ActionWorkflow®
6.3 OTHER USER ORIENTED APPROACHES
   6.3.1 Joint application development
   6.3.2 Prototyping
   6.3.3 Domain-oriented visual programming

7 THE DOMAIN LANGUAGE
    7.1 INTRODUCTION
    7.2 METAMODEL
    7.3 THE PROCESS OVERVIEW SUB-LANGUAGE
       7.3.1 Introduction
       7.3.2 The process overview language
       7.3.3 Metamodel
       7.3.4 Method
       7.3.5 Example
    7.4 THE REQUIREMENT, GOAL, PROBLEM AND MEAN SUB-LANGUAGE
       7.4.1 Introduction
       7.4.2 The RGPM language
       7.4.3 Metamodel
       7.4.4 Method
       7.4.5 Example, Fire planning
    7.5 THE PROCESS SUB-LANGUAGE
       7.5.1 Introduction
       7.5.2 The process language
       7.5.3 Metamodel
       7.5.4 Views
       7.5.5 Method
       7.5.6 Comments
       7.5.7 Example, Fire planning
    7.6 THE MESSAGE SEQUENCE SUB-LANGUAGE
       7.6.1 Introduction
       7.6.2 The message sequence language
       7.6.3 Metamodel
       7.6.4 Example
    7.7 THE ROLE SUB-LANGUAGE
       7.7.1 The role language
       7.7.2 Metamodel
       7.7.3 Example
    7.8 THE ORGANIZATION STRUCTURE SUB-LANGUAGE
       7.8.1 Introduction
       7.8.2 The organization structure language
       7.8.3 Metamodel
       7.8.4 Example
    7.9 INTEGRATED METAMODEL
    7.10 INTER-MODEL RELATIONS
    7.11 TABLES
       7.11.1 The message content table
       7.11.2 The responsibility table
       7.11.3 The role table
       7.11.4 The goal table
    7.12 METHOD
       7.12.1 Modeling of the current state
8 CASE STUDY.................................................................................................................. 145

8.1 CURRENT STATE........................................................................................................... 145
  8.1.1 Organization structure models .................................................................................. 145
  8.1.2 Role models ............................................................................................................. 147
  8.1.3 Process overview model .......................................................................................... 147
  8.1.4 Process models ......................................................................................................... 148
  8.1.5 Message sequence models ....................................................................................... 152
  8.1.6 Rule tables ............................................................................................................... 153
  8.1.7 Responsibility table ................................................................................................ 156
  8.1.8 Message content tables .......................................................................................... 157
  8.1.9 RGPM models ......................................................................................................... 160
  8.1.10 Goal Table ............................................................................................................. 160

8.2 DESIRED FUTURE STATE............................................................................................ 161
  8.2.1 Major structural changes ......................................................................................... 161
  8.2.2 Process overview model .......................................................................................... 161
  8.2.3 RGPM models ......................................................................................................... 162
  8.2.4 Goal table ............................................................................................................... 165
  8.2.5 Message sequence model ....................................................................................... 165
  8.2.6 Process models ......................................................................................................... 167
  8.2.7 Rule table ................................................................................................................ 170
  8.2.8 Organization structure models .............................................................................. 173
  8.2.9 Role models ............................................................................................................. 174
  8.2.10 Responsibility table ............................................................................................... 175
  8.2.11 Message content tables ....................................................................................... 176

9 EVALUATION.................................................................................................................... 181

9.1 ADDITIONAL USAGE...................................................................................................... 182
9.2 TOOL SUPPORT............................................................................................................. 182

9.3 CONNECTIONS BETWEEN OUR DOMAIN LANGUAGE AND DOMAIN INDEPENDENT MODELING LANGUAGES.............................................................................. 183
  9.3.1 Object oriented analysis .......................................................................................... 184
  9.3.2 Structured analysis .................................................................................................. 184

10 CONCLUSION.................................................................................................................... 187

10.1 RESULTS....................................................................................................................... 187
10.2 FURTHER WORK.......................................................................................................... 188
  10.2.1 Tool support ......................................................................................................... 188
  10.2.2 Domain language environment .............................................................................. 188
  10.2.3 Transformation of domain models to IS models .................................................... 189
  10.2.4 Domain languages used in domain analysis approaches ..................................... 189

BIBLIOGRAPHY................................................................................................................... 191

ABBREVIATIONS.................................................................................................................. 201
1 Introduction

1.1 Problem description

It is widely recognized that deep domain knowledge as well as good communication between the users\footnote{By the term user we mean domain expert, ie a person knowledgeable in the domain of interest and that is using the services and products offered by the information system.} and the information system engineers (IS engineers) in the requirements engineering stage are vital in ensuring that an information system will be in accordance with the users' needs, and developed within budget and time schedule (eg [Brown92, Curtis88, Jarke93, Miller93, Sølvberg93]).

Communication gap

However, studies show that contemporary modeling languages have not removed all communication problems between users and IS engineers [Curtis88, Miller93]. Users tend to specify their desired information system in a natural language form while IS engineers prefer to specify information systems in domain independent IS modeling languages (see section 6.2)\footnote{In the sequel "domain independent IS modeling languages" are meant by the term "IS modeling languages".}. Thus, IS engineers have to translate the users' operational requirements specification from natural language to one or more IS modeling languages, and the users have to verify the translation and validate that the IS engineers understand the problem properly.

This is a too heavy burden to both users and IS engineers. It is almost impossible for IS engineers to agree on and correctly translate a natural language specification to an IS language specification, and it is hard for users without IS engineering education to verify and validate IS models. Small IS models are mostly understandable for the users, but as the IS models grow they tend to become very difficult to understand and therefore to verify and validate. Thus, there is a communication gap between the natural language used by the users and the IS modeling languages used by the IS engineers.

The traditional communication problem between users and IS engineers is shown in Figure 1.1. Users and IS engineers all have different mental models of the world because of the differences in background, eg education and experiences. They also have different languages. Every Norwegian user and IS engineer speaks Norwegian, but they may use
different words and have different understanding of terms, and they most certainly have different professional languages due to different education and experiences. They may of course have some language in common, for example the organization chart language.

![Diagram of traditional communication between users and IS engineers.](image)

**Figure 1.1:** Traditional communication between users and IS engineers.

**Example 1**

How would a user describe or model the following situation:

- students take courses and have exams
- teachers teach courses.

Would she/he make an IS model like the one shown in Figure 1.2? We don’t think so, but many IS engineers would most certainly have made such a model.

![ER-model example](image)

**Figure 1.2:** ER-model example.
Example 2
When a military branch needs a new information system, an operational requirements specification has to be developed. The operational requirements specification has three purposes:

- force the applicants (the users) to think through their operational needs and operational consequences of a procurement
- convince decision makers (the Chief of Staff) that the needs are real and that the procurement should be given priority
- serve as the initial input to the requirements engineering stage [Pohl93].

Common problems with operational requirements specifications are that they are written in a natural language, voluminous and that they contain few drawings and models. This make them difficult to comprehend and understand for the Chief of Staff, as well as for the IS engineers.

Domain knowledge
Studies also show that IS engineers too often have insufficient domain knowledge [Curtis88, Finkelstein89, Lubars933].

"System engineer: Writing code isn’t the problem, understanding the problem is the problem." [Curtis88].

It is not possible for IS engineers to understand the problem and to communicate properly with users if they don’t have sufficient domain knowledge. Thus, it is not possible to develop an information system which is in accordance with the users’ needs, and within budget and time schedule, without sufficient domain knowledge among IS engineers.

If we for example want to develop a command, control and information system for the artillery, we have to understand the operational concepts, operational procedures, working processes, etc of the artillery, ie we have to understand the artillery domain.

"Designing from deep knowledge of the customer is central to any effective requirements definition process. ..." [Holtzblatt95]

Model dominance
Another problem faced by users participating in IS development is that IS engineers are model-strong while they are model-weak. Users are normally not used to think and reason through graphical modeling languages.

Fougere [Fougere91] states that there must be more interaction between users and IS engineers, but according to Bråten [Bråten73] isolated strategies of participation through

---

3 In the field study conducted by Lubars et al [Lubars93] lack of domain knowledge was observed in market-driven projects but not in customer-driven projects. Customer-driven projects are project "that have a single customer" while market-driven projects are projects "that have more than one actual or potential customer" [Lubars93].
communication may increase the influence gap (i.e., power gap), even if the purpose is to
the opposite, because model-strong actors may gradually increase their control of model-
weak actors. Thus, power gap in terms of model resources is critical. Unless the
participating actors have sufficient independent modeling capacity, improved
communication may actually strengthen the positions of the model-strong actors.

To day, IS engineers are in power through their IS models.

1.2 Motivation

We propose to improve on this imbalance by introducing domain languages, i.e., languages
that:

- are comprehensible to both users and IS engineers
- are able to express how the domain works
- enable IS engineers and users to have sufficient understanding of the domain
- dramatically reduce the communication problems between IS engineers and users.

To be able to meet these intentions a domain language has to be:

- based on the users’ professional language
- developed on the behalf of the users (i.e., domain experts)
- closer related to traditional IS modeling languages than natural languages are.

The main focus of a domain language is the user domain it is meant to be used to model
and not the IS domain (Figure 1.3), as is the case for IS modeling languages.

![Figure 1.3: The focus of a domain language.](image)

A domain language should be used by the users as a common representation language to
analyze and model the domain and thus to express their expectations. The use of the
domain language will enable the users and the IS engineers to develop common views and
understanding of the domain and the information system to be built.

Domain languages should typically be used in the stages prior to the requirements
engineering stage, but may also be used early in the requirements engineering stage (see
chapter 4).
1.3 Goal

We have chosen to investigate the domain of military command, control and information systems (CCIS).

Our goal is

to develop a domain modeling language for military command, control and information systems.

1.4 Approach

We have developed the domain language in close cooperation with the Norwegian artillery. Thus every example presented in this thesis is related to the Norwegian artillery. The developed domain language is firmly based on practical usefulness and it is based on a domain formalism that is a standard among the military.

The approach is based on the method engineering approach [Brinkkemper96, Kumar92, Punter96]. We have constructed a situational dependent modeling language (ie domain language) by combining ideas of well known modeling techniques with domain specific graphical symbols and terminology.

The sub-languages and tables which constitute the domain language are selected and designed based on the most important characteristics and aspects of the command and control part of CCIS.

1.5 Major contributions

The main contribution of this thesis is the development of a domain language for military command, control and information systems. In more details the contributions are:

+ the idea of using domain specific constructs in a modeling language.
+ the combination and integration of sub-languages in such a way that the compound language is especially suited for describing a particular domain.
+ the process overview sub-language.
+ the RGPM sub-language.
+ the process sub-language. It is based on DFD [Gane79], but major adjustments are done. In addition appropriate domain specific constructs are added to extend its expressive power.
the message sequence sub-language. It is inspired by CCITT [CCITT120] and the scenario language in OOram [Reenskaug96].

• the role sub-language. It is based on the traditional organization chart language, but this way of using it is the author's idea.

• the organization structure sub-language. It is based on the traditional organization chart language used by the Norwegian Army, but the addition of cardinalities and the standardized way to draw the organization charts are the author's idea. The idea of integrating the organization structure language with other modeling language is the author's too.

• the domain oriented tables.

1.6 Outline of the thesis

The organization of the thesis is as follows:

• In chapter 2 characteristics of command, control and information system are presented along with a short description of the contemporary development approach for CCIS'.

• Chapter 3 introduces military symbols for land based systems and the military symbols used in our domain language.

• In chapter 4 we explain where in the information system development life cycle the domain modeling should be carried out.

• Requirements to our domain language are stated in chapter 5.

• In chapter 6 some domain dependent modeling languages are presented along with some domain independent modeling languages and some other user oriented approaches.

• Our domain language is presented in chapter 7.

• An almost complete modeling example from the domain of artillery is presented in chapter 8.

• In chapter 9 our contribution, ie the domain language, is discussed along with a discussion on tool support and how domain independent modeling languages could be connected to our domain language.

• Finally, chapter 10 concludes the thesis.
2 Command, Control and Information Systems

2.1 Introduction

The term **command, control and information system** (CCIS) includes a very wide variety of information systems. However, this thesis is restricted to military CCIS'.

In [AAP-6] the term **command, control and information system** is defined as:

An integrated system comprised of doctrine, procedures, organizational structure, personnel, equipment, facilities and communications which provides authorities at all levels with timely and adequate data to plan, direct and control their activities.

This definition is all-including. It includes information (data), everything needed to collect data (doctrine, procedures, organizational structure, personnel, equipment and facilities), information conveyance (communications), command (plan and direct) and control. From the definition of CCIS we can conclude that CCIS', like other information systems, consist of manual parts, information technology (IT) supported parts and automated parts. In the sequel IT supported parts and automated parts will be called computerized parts.

A CCIS could be divided into three parts. The information collection part, the command and control (C&C) part, and the action part as shown in Figure 2.1.

![Diagram](image)

Figure 2.1: CCIS divided into three parts.

The command and control part typically covers the headquarters at the company/-
battery/squadron level⁴ and above (Figure 2.2).

Figure 2.2: CCIS hierarchy.

Figure 2.3 gives a pictorial view of our understanding of the term command, control and information system. Information is collected from the battle field and sent to the command and control part of the CCIS. Based on the information from the battle field and the information in the information storage decisions are made and orders diffused.

Figure 2.3: Command, control and information system.

⁴ See subsection 3.1.4.
We have adopted the following understanding of the terms \textit{command} and \textit{control} [Bourdin94]:

- \textit{command} addresses mission planning, forces and resources allotment, analyzing, and task orders diffusion
- \textit{control} addresses information acquisition, situation monitoring, operations and situation analysis, forces and resources management, and follow-up of the correct execution of orders.

The information storage contains information about our own forces, their resources, capabilities, combat power, logistics, orders and other necessary information, for example firing data and map data. The information storage also contains information about the enemy forces, their resources and capabilities, as well as information collected from the battle field, for example where our own and the enemy units are located.

Lawson's model of the command and control process [Lawson81, Bourdin94] has five steps as shown in Figure 2.4: 1) Information about the adversary, the environment and own forces are reported to the command and control center by sensors and own forces. 2) This information together with information from cooperating centers (external data) is processed to give an overview of the situation on the battle field. 3) By comparing the desired situation with the current situation, 4) the decision makers can decide what to do next. 5) The wanted actions are transmitted to own forces as orders.

In this model it seems that the command and control center is treated as being located outside of the battle field area. In practice, command and control centers are indeed not located at the front line but they are of course part of the battle field area and they are very attractive targets.

![Diagram of command and control process](image_url)

\textbf{Figure 2.4:} The Lawson model of the command and control process.

Colloquially it is the command and control part which is meant by the term command,
control and information system, ie we mean the command and control information system when we talk about the command, control and information system. It is the command and control part that constitute the lion's share of CCIS' and that is the most wicked one (the properties of wicked problems are discussed in [Sølvberg93]).

Thus, our domain language should be tailored to the command and control part of the CCIS. In the next section characteristics of command and control information systems (C&CIS) are described.

2.2 Characteristics of C&CIS'

C&CIS', and other decision oriented systems, may be viewed as a network of decisions centers (Figure 2.2). Each center receives, processes and dispatches a large amount of information, and the process involved in each decision center adds value to the initial information [Bories91].

A C&CIS shall provide decision makers at all levels with timely and adequate information to give them freedom of action and decision support. The computerized part of a C&CIS may be viewed as a special type of executive information system (EIS) [Watson93, Leidner93] as illustrated in Figure 2.5.

```
EIS
/ is-an
Computerized CCIS
```

Figure 2.5: The computerized part of a C&CIS is an EIS.

The battle field is the operational environment of military C&CIS'. It is a stressful, tough, high-velocity environment where continuous maneuvering and rapid and continuous changes in own and enemy forces (ie resources, capabilities and combat power) are the rule and not the exception. To be less vulnerable the trend is many, small, mobile and distributed units.

It is important that C&CIS' continuously keep track of the situation. A commander must at any time know what resources he has and what resources the enemy has, where they are located, their capabilities and their combat power. It is a question of life or death.

"Know the enemy and know yourself; in a hundred battles you will never be in peril. When you are ignorant of the enemy but know yourself, you chances of winning or losing are equal. If ignorant both of your enemy and of yourself, you
are certain in every battle to be in peril." - Sun Tzu, 500 bc⁵ [Griffith'71].

In such an operational environment it is very important:

- to have timely and adequate information
- to be able to receive and systematize the large amount of information that is terminated in the headquarters
- to have a decision support system
- to have a superb presentation system which gives a superb overview of the situation. To be able to plan, direct and control military operations it is very important to have an updated and comprehensible presentation of the situation
- to have efficient and effective working processes. Important principles are simplicity and similarity. Simplicity and similarity are especially important in a country, eg Norway, where the defense are based on reserve officers and conscripts. If a language with very high expressive power is needed to describe a process, the process is probably to complex and should be redesigned.
- to delegate. Delegation is important because the response times might be lower and the flexibility might be higher if the delegation is done in a proper way. Delegation is composed of three elements: task⁶, responsibility⁷ and authority⁸. Successful delegation depends on among others the parity principle, ie the person who has the responsibility to perform a task should have equivalent authority to perform it [Hitt89]. The parity principle is very important, but often sinned against.
- to have clear responsibilities, ie everyone must know what to do and everyone must know who is responsible for doing what
- for a commander to know his own and the enemy's organization, their resources, capabilities and combat power
- to have optimal resource management.

In these aspects C&CIS' are not very different from other decision oriented systems. Every organization is striving to reach the points listed above. The main differences are the time aspect, ie available time to make decisions (the decision window), and the deadly environment. You have to make the right decision fast the first time. If you are just second best you are dead. Therefore, it is of utmost importance that IS engineers and users have a sufficient understanding of the CCIS domain and that IS engineers and users are able to communicate with each other without misunderstandings.

Special characteristics of military C&CIS, especially the computerized part, are said to be security, reliability, interoperability and survivability [AC/243], but these characteristics are valid for most computerized information systems. We believe that the most special characteristics of C&CIS' are redundancy and transmission rate.

Redundancy is part of the survivability and it means that the same information is stored in more than one headquarters. It is a kind of back-up in case of degradation (ie loss of

---

⁵ BC: Before Christ.
⁶ Task is the activities to be accomplished.
⁷ Responsibility is the requirement of a person to perform assigned tasks.
⁸ Authority is the right to take action and use resources.
headquarters) as depicted in Figure 2.6. In Figure 2.6 the artillery battalion is destroyed and its duties are immediately taken over by one of its batteries.

The key questions are which headquarters should be back-up for which and how many back-up headquarters should there be. These questions are among others influenced by the transmission rate.

![Diagram](image)

Figure 2.6: Redundancy.

The transmission rate of tactical military communication systems is very low compared with commercial communication systems. Special attention should therefore be paid to transmission routines and transmission protocols.

2.3 Contemporary development approach for CCIS'

The content of this section is based on the authors own experiences as a project leader for a C&CIS project in The Norwegian Army Material Command and on the project manual [Forsvaret95] used by the Norwegian armed forces.

The project manual [Forsvaret95] introduces a project model consisting of five phases: concept, definition, development, acquisition, and transfer. For each phase a lot of stages are prescribed.

The concept phase consists of 13 stages. One stage is concerned with the preparation of a goal document containing objective settings and another stage is concerned with the preparation of the first version of a requirements specification describing operational requirements. These two documents are usually prepared by the users in collaboration with the material command. The operational requirements specification is further elaborated on by the users, in collaboration with the material command, in the definition phase. This elaboration is based on among others a system study accomplished by the material command in the definition phase. The Norwegian Defense Research Establishment, industry or consultants may be involved in the preparation of the system study which may include development of different kinds of prototypes. The operational requirements specification, the goal document and the system study are important inputs to the development phase which among others includes a stage concerned with the
preparation of a requirements specification describing functional and non-functional software requirements. The requirements specification is usually accomplished by industry or a consultant company in collaboration with the users and the material command.

The author's experiences with the operational requirements specification and the goal document are that they are ambiguous, incomplete, inconsistent, written in natural languages, voluminous and that they contain few drawings and models. Thus they are difficult to understand for the decision makers, as well as for IS engineers.

Users and the IS engineers or action officers (clerks) in the material command communicate mostly by natural language, written and oral, but also the operational language based on military symbols and organization charts are used. In Norway the IS engineers in among others the army material command have started to make behavior models in the concept phase and the definition phase to improve on the knowledge acquisition process, but the used behavior modeling language (the Behavior Graph Notation used in RDD™ (Requirement Driven Developer) [Ballard89]) is not domain oriented and it is not used by the users.

The communication between users and IS engineers from industry/consultant companies is mostly based on a combination of natural language, graphical models and prototypes, where the users mostly are using natural language, their operational language based on military symbols, and organization charts.

It is the author's experience that the communication between users and IS engineers, both IS engineers from the material command and from industry/consultant companies, is hampered by the use of different languages (natural language vs modeling languages) and the IS engineers lack of domain knowledge and domain terminology. Due to the IS engineers lack of domain knowledge much of a project's time is spent in learning. This knowledge acquisition process is necessary and useful, but the bad thing is that the same "lecture" is given two, three or four times. First: the users have to educate IS engineers in the material command during the concept and definition phases, second: consultants involved in the system study in the definition phase have to be educated, then the consultants responsible for writing the requirements specification have to be educated, and fourth: a new vendor might be selected to design, implement and produce the system and these people also have to learn the domain, and worse they have to read and grasp a requirements specification written by others. This is a poor procurement practice, which also is discussed in [AC/243].

We recommend the use of a domain modeling language, as defined in chapter 7, to reduce the communication gap between users and IS engineers. We believe that the use of a domain language will improve the domain knowledge acquisition process and thus enable users and IS engineers to faster reach a common understanding of the system to be developed. We believe it will be beneficial for the users too to use a common modeling language to describe their domain. The use of a domain language will enable the users to develop common views and understanding of the domain and the information

---

™ RDD is a trademark of Ascent Logic Corporation.
system to be built. The use of a domain language may also enable them (the users) to think and reason through graphical models and thus grasp other models, eg domain independent models, faster.
3 Military Symbols for Land Based Systems

In this chapter we present the professional language used by the military.

The first section of this chapter gives an general introduction to military symbols for land based systems while the second section introduces the military symbols used in our domain language for CCIS'.

3.1 Introduction to military symbols for land based systems

This section gives an introduction to military symbols for land based systems. The introduction is mostly based on APP-6 [APP-6] which is a NATO unclassified publication.

Most of the symbols in APP-6 and some specialized symbols which are not included in APP-6 are part of the professional language used by the Norwegian artillery. A specialized terminology, with a lot of abbreviations, is also included in that language.

A military symbol is defined as:

“A graphic sign used, usually on a map, display or diagram to represent a particular military unit, installation, activity or other item of military interest. It may be in colour and is normally accompanied by alphanumeric characters” [APP-6].

3.1.1 Color presentation

Different colors are normally used to differentiate between enemy and friendly military units, posts, installations, equipment and activities. Blue or black is used to denote friends and red is used to denote enemies. If it is just possible to use one color, different line types are used to differentiate between enemies and friends. Single lines are used to denote friends while double lines are used to denote enemies, as shown in Figure 3.1.
Figure 3.1: One color presentation of friends and enemies.

3.1.2 Basic symbols

There are seven basic symbols in APP-6, as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle</td>
<td>units.</td>
</tr>
<tr>
<td>Rectangles with a shaft</td>
<td>headquarters (HQ) or an element of a headquarters.</td>
</tr>
<tr>
<td>Equilateral triangle</td>
<td>observation posts.</td>
</tr>
<tr>
<td>Circle</td>
<td>logistic or administrative installations.</td>
</tr>
<tr>
<td>Up side down equilateral triangle</td>
<td>electronic installations.</td>
</tr>
<tr>
<td>Wing</td>
<td>landing sites.</td>
</tr>
<tr>
<td>Rectangle with a broken line above</td>
<td>forces under national command and not assigned to NATO.</td>
</tr>
</tbody>
</table>

Table 3.1: Basic symbols.

Note:

1. Normally a unit is composed of (ie an aggregation of) several other units, headquarters, observation posts and/or installations as shown in Figure 3.2. It depends on what level one is observing.
2. The term unit may be used to denote any military element, eg observation posts and installations are all military units (Figure 3.2).
3. Observation posts and installations are not composed of other military units (Figure 3.2).
4. HQ are composed of centers (Figure 3.2).
5. Headquarters (HQ) may also be denoted command post (CP), ie a CP is a HQ. The term CP are normally used to denote small headquarters.

If the type of unit, HQ, observation post or installation is known, the relevant basic symbol is filled with a role indicator. A role indicator is an icon and it is either basic or
composite. A composite role indicator is composed of two or more basic role indicators.

Figure 3.2: Unit.

Role indicators
Some of the basic role indicators placed within units and headquarters, and included in APP-6, are shown in Figure 3.3. Note: in the sequel we mean field artillery when the term artillery is used. Artillery is the short form of the term field artillery. The term field artillery is abbreviated FA.

Figure 3.3: Basic role indicators, unit and headquarters.
Some of the composite role indicators placed within units and headquarters, and included in APP-6, are shown in Figure 3.4.

Figure 3.4: Composite role indicators, unit and headquarters.

Some specialized composite role indicators used in Norway are shown in Figure 3.5, i.e., these role indicators are not included in APP-6.

Figure 3.5: Specialized role indicators, unit and headquarters.

Two kind of observation posts are used in the Norwegian artillery, forward observers and static forward observers. They are shown in Figure 3.6.

Figure 3.6: Observation posts.

Some of the composite role indicators placed within logistic installations, and included in APP-6, are shown in Figure 3.7.

Figure 3.7: Composite role indicators, logistic installation.
One composite role indicator placed within electronic installations are shown in Figure 3.8. This composite role indicator is not included in APP-6.

![Artillery locating radar](image)

Figure 3.8: Composite role indicators, electronic installation.

### 3.1.3 Equipment symbols

Two of the weapon equipment symbols included in APP-6 are shown in Figure 3.9.

![Medium gun/Howitzer](image) ![Light gun/Howitzer](image)

Figure 3.9: Weapon equipment symbols.

### 3.1.4 Fields

Basic symbols and equipment symbols are surrounded by lines that provide additional information. Each line consists of one or more fields. The position and length of these fields are regulated through strict rules so that the map is not cluttered with too much additional information. Most of the fields are optional.

Each field has the following characteristics:

1. Field title. The name given to a particular position in relation to the symbol, e.g. “unique designation”.
2. Field length. The maximum number of alphanumeric characters which may be used in a particular field.
3. Field letter. The position of each field in relation to the symbol is shown on a field diagram by means of a letter which stands for the field title.

The position of each field, the field letter and the maximum length of each line⁹ are shown in Figure 3.10.

---

⁹The line length is the maximum number of alphanumeric characters which may be used in a particular line.
Figure 3.10: Field diagram.

Note:

1. to avoid confusion the letters "I", "O" and "U" are not used to indicate fields
2. the field E has a maximum field length of one even though the maximum line length is 21
3. the order of the fields H/J/K/L are not logical because the field H should always be last when a combination of those fields are used
4. alphanumeric characters are not used in every field even though the maximum field length is given in numbers of alphanumeric characters
5. the font type and font size of the alphanumeric characters are not given.

The usage of the fields may depend upon the role indicator, the equipment symbol, or if it is a friendly or enemy unit.

Mandatory fields must be completed if possible. If the information is not known, a query mark must be shown in its place. If the information is doubtful, a query mark must be shown at the end of the information. This is shown in Figure 3.11.

Figure 3.11: Unknown or doubtful information.

The field titles, the field lengths (shown in brackets), and the field contents are:

A - Role indicator (-). Mandatory for basic symbols. A role indicator is an icon, and it is either basic or composite. A role indicator is placed within a basic symbol. It is used to show the type of unit, headquarters, observation post or installation. A composite role indicator is composed of two or more basic role indicators.

B - Size indicator (-). Mandatory for units and headquarters, but might also be used with some installations and equipment. The size indicator is placed on top of units, HQ, installations and equipment symbols to show their size. The following size indicators are commonly used in the Norwegian Army\(^\text{10}\), where *squad* is the smallest size (Figure 3.12):

\(^{10}\) Group: In Norway the term "Group" is used to denote the unit which in APP-6 is called "Unit larger than a Squad but smaller than a Platoon".
Figure 3.12: Size indicators.

Note that a squad is part of a group, a group is part of a platoon, and so on. It is an organizational hierarchy.

C - Quantity of equipment (6). Optional. This field is used with equipment symbols only. It is used to show the number of an equipment. Figure 3.13 displays two medium guns.

2

Figure 3.13: Quantity of equipment.

D - Special size indicator (-). Mandatory for units and installations if the conditions which the field indicates are applicable, for example a special grouping of units. Figure 3.14 displays an infantry unit in special grouping, for example a combat team.

Figure 3.14: Special size indicator.

E - Unconfirmed or query mark (1). Mandatory when all the information is doubtful or unconfirmed. A "?" is shown in field E if all the information is doubtful or unconfirmed. The unit displayed in Figure 3.15 is thought to be an armored artillery battalion, but all the information is unconfirmed.
Figure 3.15: Unconfirmed or query mark.

F - Reinforced or detached (3). Optional. Mostly used with units. This field is used to indicate if a unit is reinforced "(+)" or if parts of a unit are detached "(-)". Brackets are always used. Figure 3.16 displays that the 1st Infantry Battalion is reinforced and that the 2nd Infantry Battalion is detached.

Figure 3.16: Reinforced and detached infantry battalions.

G - Additional information (17). Optional. Space for notes.
H - Free text (15). Optional. Space for additional information not covered by the other fields.
J - Evaluation rating (2). Optional. Used with enemy forces only. It consists of one letter to show the source grading and one figure to show the information grading. This is displayed in Figure 3.17.

Figure 3.17: Evaluation rating.

K - Combat effectiveness (5). Optional. Gives an indication of the combat effectiveness of a unit; friendly or enemy. It may be shown as:

- combat effectiveness, ie CE plus two figures
- percentage effectiveness, ie PC plus two figures, as shown in Figure 3.18.

Figure 3.18: Combat effectiveness shown as percentage effectiveness.

L - Signature equipment (1). Optional. Used with enemy equipment only. The character "!" is used if equipment emit a signature.
M - Higher formation (15). Optional. This field is used to show some or all of a unit's, headquarters', installation's, observation post's or equipment's higher units or formations. Higher units are separated by slashes. Abbreviated titles are always used, and national distinguishing letters, in brackets, are used if necessary. If a command level is unused, the symbol "Ø" must be shown. Figure 3.19 tells that the Mike US Armored Artillery Battery is part of the NCF US Armored Artillery Battalion (NCF: NATO Composite Force) which is part of the 1st Norwegian Armored Artillery Regiment.

![Figure 3.19: Higher formations.](image)

N - Enemy (2) (shown as "EN"). Mandatory for enemy equipment if one color representation is used. Figure 3.20 displays an enemy medium gun.

![Figure 3.20: Enemy.](image)

P - Addressing number (5). Mandatory for enemy forces. It is used when the identification is incomplete or doubtful. The addressing number is always shown in brackets to distinguish it from the M field - Higher formation. Figure 3.21 displays an enemy unit which is given the addressing number 123.

![Figure 3.21: Addressing number.](image)

Q - Direction of movement arrow (-). Optional. An arrow is used to give an indication of the direction of movement. Figure 3.22 displays that the Mike Artillery battery is moving west and that an artillery battalion headquarters is moving east.
R - Mobility indicator (-). Optional. For units and equipment only. A pictorial representation of the type of mobility, for example:

- Wheeled
- Wheeled, cross country
- Tracked

Figure 3.23 displays an infantry company in wheeled transport.

S - Headquarters representation (6). Optional. For headquarters only. An abbreviation is used to show the type of headquarters.

T - Unique designation (15). Mandatory for units and headquarters. This field gives the title of a unit or headquarters. For installations, observation posts and equipment it shows the owner, ie which unit the installation, observation post or equipment belongs to. Abbreviated titles are always used, and national distinguishing letters, in brackets, are used if necessary. The unique designation must always match the size indicator. Figure 3.24 displays the Mike Artillery Battery.

V - Name of unit or type of equipment (15). Optional. This field is used to show the name of the unit, installation or equipment rather that its unique title. Figure 3.25 tells that the B Infantry Company is a special force unit.
3.1.5 Examples showing composite use of some fields

1. Figure 3.27: The C Infantry Company, 14th Infantry Battalion of the 12th UK Brigade is being used as a patrol company. The C Company is detached (ie just part of the C Company is used as a patrol company. Note that UK does not use “regiment”, so an “Ø” is shown.

2. Figure 3.28: The headquarters of the 2nd Artillery Battalion, 3rd Regiment, 76th Brigade of the 2nd US Division is about to move east, but “no move before” (NMB) 1100 o’clock Zulu time (Z) on the 3rd of the current month. The information was valid at 1005 Zulu time (Z) on the 3rd of the month.
3. Figure 3.29: The enemy's 6th Tank Regiment, equipped with T81 tanks, was observed at 1500 Zulu time (Z) on the 10th of the month. The information is graded as "B1". The tank regiment is given the addressing number 231 because the information is incomplete and doubtful.

![Figure 3.29: Enemy tank regiment.]

4. Figure 3.30: Two 155 mm guns belonging to the Mike Artillery Battery, 1st Artillery Battalion of the 1st Artillery Regiment.

![Figure 3.30: 155 mm gun.]

5. Figure 3.31: An artillery ammunition point manned by the Papa Artillery Battery, 3rd Artillery Battalion of the 15th Brigade will open at 0500Z on the 3rd day of the month. The exact location of the ammunition point is indicated by the vector.

![Figure 3.31: Artillery ammunition point.]

6. Figure 3.32: A forward observer located at ground coordinates (GR) 965495 from 1500 Alfa time (A) on the 3rd day of the month to 2300A on the 6th day of the current month. The forward observer belongs to the 1st Artillery Regiment.

![Figure 3.32: Forward observer.]
3.2 Military symbols used in our domain language

We are not going to use every military symbol and every field defined in APP-6 in our domain language for military CCIS.

3.2.1 Basic symbols

Four of the seven basic symbols are included in our domain language. The four basic symbols are shown in Table 3.2.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle</td>
<td>units, headquarters (HQ)/command posts (CP) and commanders.</td>
</tr>
<tr>
<td>Equilateral triangle</td>
<td>observation posts.</td>
</tr>
<tr>
<td>Circle</td>
<td>logistic or administrative installations.</td>
</tr>
<tr>
<td>Up side down equilateral triangle</td>
<td>electronic installations.</td>
</tr>
</tbody>
</table>

Table 3.2: Basic symbols.

In APP-6 a unit is represented by a rectangle, a headquarters or an element of a headquarters is represented by a rectangle with a shaft, while commanders are not represented by a particular military symbol. In our domain language rectangles are used to represent all these three concepts: units, headquarters and commanders.

We have chosen to include commanders in our domain language because commanders are often on the move and therefore not in the HQ, but they still need to be kept informed about the situation.

We could have chosen to represent these three concepts by three different graphical symbols. Our domain language would then have been extended with two more graphical symbols without an equivalent extension in expressive power or comprehensibility of the models made by the language. The reasons are:

• commanders are identified by unique abbreviations. Thus, people who knows the abbreviations are able to distinguish commanders from units and headquarters
• the context will distinguish units from headquarters.
3.2.2 Equipment symbols

The equipment symbols used in APP-6 or specialized equipment symbols used by the Norwegian army could all be used in our domain language.

3.2.3 Fields

We use five of the 21 fields in our domain language, one field is mandatory and four fields are optional. These five fields are shown in Figure 3.33.

![Field diagram](#)

**Figure 3.33: Field diagram.**

A - Role indicator (mandatory)
B - Size indicator.
G - Additional information.
M - Higher formation.
V - Name of unit or type of equipment.

Field G is used to show a particular center in a headquarters, eg the fire support coordination center (FSCC) in a brigade (Figure 3.34 a)), while field V is used to show that a unit or headquarters has been assigned a special role, eg a reinforced artillery battalion, Rbn (Figure 3.34 b)).

![Field diagram](#)

**Figure 3.34: a) Fire support coordination center, b) Reinforced artillery battalion.**

Field M is used if the same type of unit is included in two or more different higher units. Field M indicates the higher unit that the unit is part of. Figure 3.35 show that a signal platoon is included in both field artillery staff batteries (FA-Staffbtt) and field artillery batteries (FAbtt).

![Field diagram](#)

**Figure 3.35: Field M, higher units.**
4 Domain Redesign and Requirements Engineering

4.1 Introduction

We will in this chapter present an information system life cycle model for understanding where in the IS life cycle the domain modeling is carried out.

In Figure 4.1 four stages of an information system life cycle model used by IFIP Working Group 8.1 are shown [Olle91].

![Diagram](image.png)

Figure 4.1: Part of the information system life cycle model used by IFIP Working Group 8.1.

The stages are:

1. **Information systems (IS) planning.** Most enterprises require a substantial number of computerized information systems (CIS). The number is so great that it is impossible to develop all at the same time. Information systems planning is therefore concerned with:
   
   - analyzing the enterprise’s use and needs of information
   - analyzing which computerized information system will gain the enterprise most
   - decide which computerized information system should be developed when.
2. **Business analysis.** This stage is concerned with analyzing the existing state of affairs in a given business area of the enterprise.

3. **System design.** The goal of this stage is to build requirements specifications of the computerized information systems.

4. **Construction design.** The goal of this stage is to build an information system design specification of the computerized information system.

In [Olle91] the term analysis is used to describe the process of looking at something that already exists. Thus, it seems that this information system life cycle model is not concerned with necessary organizational changes, eg business process redesign and organizational development.

We prefer an information system life cycle model which in addition to the information systems development considers necessary organizational changes. We want to explicit show that information systems development and organizational changes are two of a kind, ie it is normally not a good solution to develop and install a new information system without developing the organization at the same time because information systems are integrated in the organization. Three of the stages in our information system life cycle model are shown in Figure 4.2.

![Figure 4.2: Three of the stages in our information system life cycle model.](image)

These three stages are:

1. **Enterprise analysis.** This stage is concerned with analyzing the existing state of affairs in the whole enterprise, or part of the enterprise. The enterprise analysis is mainly concerned with the value chain and how to support the value chain with aid of information technology to create competitive advantage and new businesses [Porter85]. In this stage it is decided which information system should be developed
when.

2. **Domain redesign.** The main goal of the domain redesign process is to build the users' operational requirements specification. It should include both organizational aspects and IS aspects.

3. **Requirements engineering.** The goal of this stage is, as for the system design stage in the IFIP model, to build requirements specifications of the computerized information systems.

Domain redesign must not be confused with *domain analysis* or *domain engineering* which are software oriented and reuse motivated [Tracz93, HP94, O'Connor94, Rolling94, Batory95, Davis95, DISA95, France95, Gossain95, Henninger95, Svoboda96].

Domain analysis/engineering aims at preparing reuse by centering the analysis on identifying common and variable parts of software systems in a particular domain.

"*Domain engineering describes the work required to define, specify, and implement a software product line*" [Macala95].

"*Domain analysis, the activity of identifying and documenting the commonalities and variabilities in a set of related software systems*" [Frakes95].

Some domain analysis/engineering approaches focus on identifying common and invariant parts while other focuses on identifying common and variant parts.

Domain redesign must neither be confused with *domain knowledge modeling* which is a reuse motivated approach used in connection with development of knowledge-based software systems [Abu-Hanna94].

Domain redesign is further discussed in section 4.3 while requirements engineering is further discussed in section 4.2.

### 4.2 Requirements Engineering

The goal of the requirements engineering process is to build a requirements specification of the computerized parts of an information system.

The requirements specification should be **complete** and expressed using a **formal language**, but it is not enough to produce a complete specification expressed in a formal language. All people involved in the requirements engineering process should also agree on the specification, i.e., they should have a **common view**. This is called the "desired output" in Figure 4.3 [Pohl93].

Normally, the knowledge about the domain and the system to be developed is coarse at

---

11 Bibliography on domain engineering.
the beginning of the requirements engineering process. Most users have their own personal view of the system, they don't use a common representation format, ie language, to express their expectations, and the used representation languages are mainly informal. Therefore the understanding of the domain, the system and the specification which can be gained out of it is very opaque. This is called the "initial input" in Figure 4.3 [Pohl93]. The initial input to the requirements engineering process is often called an operational requirements specification, and it is normally built by the users them selves.

From the described characteristics of the initial input and the desired output Pohl [Pohl93] identifies three main goals of the requirements engineering process:

- improving an opaque system comprehension into a complete system specification
- transforming informal knowledge into formal representations
- gaining a common agreement on the specification out of the personal views.

Out of these goals, Pohl identifies three dimensions of the requirements engineering process:

- the specification dimension
- the representation dimension
- the agreement dimension.

These three dimensions are called Pohl's framework and it is shown in Figure 4.3.

![Diagram showing three dimensions of requirements engineering](image)

**Figure 4.3: The three dimensions of requirements engineering**

**The specification dimension**
The specification dimension deals with the degree of requirements understanding at a
given time. Focusing on this dimension, the aim of the requirements engineering process is to transform the users’ operational requirements into a complete requirements specification.

Several standards and guidelines describe how the final requirements specification should look like, eg IEEE Std. 830-1993 [IEEE830] and DOD-STD-2167A [DOD2167A] (both are software oriented).

The representation dimension
The representation dimension deals with the different languages used for expressing knowledge about the system. During the requirements engineering process three different categories of representation languages are used:

- informal, eg naturals languages, arbitrary graphics, case studies, etc.
- semi-formal, eg data flow diagram languages [Gane79], ER modeling languages [Chen76], PPP [Gulla91], OOMT [Rumbaugh91], OOram [Reenskaug96], and many more. In contrast to informal languages the semi-formal languages come with formally defined semantics, but the formal defined semantic of semi-formal languages is very poor, so still most of the represented knowledge has no formal meaning.
- formal, eg specification languages like TROLL [Jungclaus96], VDM++ [Dürr92] and Z [Spivey89], and knowledge representation languages like ERAE [Dubois88] and Telos [Mylopoulos90].

The representation language used does not imply if a requirements specification is vague or precise. Hence the representation dimension is orthogonal to the specification dimension. A vague imagination of the system can be expressed using a natural language, but also using a formal representation language.

The agreement dimension
The agreement dimension deals with the degree of agreement reached on a specification. At the end all people involved in the requirements engineering process should agree on the specification, ie they should have a common view.

4.3 Domain redesign

This stage is concerned with analyzing how the domain works, eg operational concepts, business processes, information flows, organization etc. If necessary the operational concepts, business processes, etc are redesigned or designed from scratch. The goal of this stage is to redesign inefficient or design non existing operational concepts, business processes, etc and to build the users' operational requirements specification, which includes both organizational aspects and IS aspects.

It is our conviction that users in a domain should use a domain language as a common representation language in the domain redesign process. The domain language should be used by users and IS engineers to document user expectations in an operational requirements specification. The use of a domain language will enable the users and the IS
engineers to have an almost common view and understanding of the domain and the information system to be built.

The main idea behind the use of domain languages is to raise the initial input to the requirements engineering process from an opaque and disagreed (personal views) requirements specification expressed in informal representation languages to a fair and almost agreed requirements specification expressed in semi-formal representation languages as illustrated in Figure 4.4.

![Diagram](image)

Figure 4.4: Raised initial input.

Domain languages should be semi-formal, ie they must have formally defined syntax and semantics. And to repeat, a domain language should be:

- based on the users' professional language
- able to express how the domain works
- developed on the behalf of the users
- closer related to traditional IS modeling languages than natural languages are.

We don't want the domain languages to be formal because formal languages are often hard to understand and therefore not suitable to be used by the users (ie domain experts) [Brown90, Davis90].

"We do not believe that it is realistic to expect that dialogue between the operational users and systems developer can be conducted in any of the presently available formal specification languages .." [Brown90].
To summarize:

- the use of a domain language aims to give the users and the IS engineers an almost **common view** of the domain and the IS system to be developed
- the domain language is a common representation format in which the users may express their expectations
- the domain language is **semi-formal**
  \[\Rightarrow\] therefore, the understanding of the domain, the system and the specification which can be gained out of it is **fair**.
5 Language Requirements

As stated in chapter 1, domain languages should:

- be comprehensible to both users and IS engineers
- be able to express how the domain works
- enable IS engineers and users to have sufficient understanding of the domain
- dramatically reduce the communication problems between IS engineers and users.

To be able to meet these intentions a domain language has to be:

- based on the users' professional language
- developed on the behalf of the users
- closer related to traditional IS modeling languages than natural languages are.

In the next section, 5.1 Language quality, we present the language quality requirements developed by Krogstie [Krogstie95]. In section 5.2 the domain language requirements are presented.

Note that most of Krogstie's requirements are intentions and not true requirements because true requirements should be measurable. Non-measurable requirements are intentions.

5.1 Language quality

The language quality requirements presented in this section are based on the work done by Krogstie [Krogstie95]. Krogstie has grouped the quality factors in four groups:

- domain appropriateness
- participant knowledge appropriateness
- participant interpretation enhancement
- technical actor interpretation enhancement.

In addition, for every group he distinguish between the underlying basis of a language and its external representation.
5.1.1 Domain appropriateness

Domain appropriateness means that there are no statements in the domain that can not be expressed in the language.

Underlying basis
The language must be powerful enough to express anything in the domain. To be able to accomplish that:

- the phenomena\(^{12}\) must be general rather than specialized
- the phenomena must be composable, i.e., it must be possible to group related statements in a natural way
- the language must be formal and unambiguous
- the language must have vague constructs for modeling vague knowledge. To be able to fulfill the previous requirement, the vagueness must be formalized
- the language must support all relevant perspectives. According to [Krogstie95] the relevant perspectives are:
  
  - the structural perspective (called the data perspective in [Yang93]), which is handled by languages for data modeling like the entity-relationship (ER) language [Chen76]
  - the functional perspective (called the process perspective in [Yang93]), which is handled by process-oriented languages like the data flow diagram language (DFD) [Gane79], the process modeling language in IE (Information Engineering) [Martin90], the Functional Model Notation in OMT (Object Modeling Technique) [Rumbaugh91], and the Process Port Model (PPM) in PPP [Yang93]
  - the behavioral perspective (called the event/behavior perspective in [Yang93]), which is handled by behavior-oriented languages like the Behavior Graph Notation used in RDD\(^{TM}\) (Requirement Driven Developer) [Ballard89], Statecharts [Harel87], Petri Nets [Reisig85], SDL (Specification and Description Language) [Breuk93] and other STD (State Transition Diagram) oriented notations
  - the rule perspective, which is handled by rule-oriented languages, e.g., Tempora [Loucopoulos91], or goal-oriented languages, e.g., the objectives world in the Esprit project “From Fuzzy to Formal” [Nellborn92]
  - the object perspective, which is handled by object-oriented languages like OMT [Rumbaugh91] and OOram (Object Oriented role analysis and modeling) [Reenskaug96]
  - the communication perspective, which is based on the speech act theory developed by Austin [Austin62] and Searle [Searl79]. The communication perspective is handled by languages like Action Workflow [Medina-Mora92], SAMPO [Auramäki92] and COMMODIOUS [Holm94]
  - the actor, agent and role perspective (called the role perspective in [Yang93]), which is handled by actor-oriented languages, e.g., the actors world in the Esprit project “From Fuzzy to Formal” [Bubenko92], agent-oriented languages, e.g.

\(^{12}\)A phenomenon could be a process, an entity, a message, a trigger, a store, an object, etc.

\(^{TM}\)RDD is a trademark of Ascent Logic Corporation.
ALBERT (Agent-oriented Language for Building and Eliciting Real-Time requirements) [Dubois94], or role oriented languages, eg OOram [Reenskaug96]. Note that OOram is both object and role oriented.

**External representation**
The external representation must not destroy the underlying basis, ie:

- every possible statement in the language should have a unique representation in the underlying basis
- every possible statement in the underlying basis should have at least one external representation (but it is all right to have several alternative external representations for the same statement)
- it must be possible to compose the symbols of the language.

### 5.1.2 Participant knowledge appropriateness

Participant knowledge appropriateness means that all the statements in a model of a language are part of the modeler’s (ie participant’s) explicit knowledge.

**Underlying basis**
The underlying basis should correspond as much as possible to the way individuals perceive reality. An empirical study performed by Vessey and Conger [Vessey94] reports that novice analysts seemed to have much greater difficulty applying an object methodology, than a data or process methodology. The study also reports that process modeling was found easier to apply than data modeling. A similar experience is reported in [Tempora94], participants had small problems in learning to use the process modeling language and the main parts of the data modeling language, while they found it difficult to comprehend the formal textual rule language.

It should be possible to express inconsistencies because inconsistency between how participants perceive reality is a fact of life which is useful to represent so that inconsistencies can be revealed and discussed explicitly. According to Feather and Fickas [Feather91] it is productive to maintain inconsistencies because conflict fosters creativity.\(^\text{13}\)

**External representation**
The external representation of the different phenomena should be intuitive in the sense that the symbol chosen for a particular phenomenon somehow reflects this better than another symbol would have done.

---

\(^\text{13}\)A cooperative approach to conflict, which emphasizes their mutual goals, may have a constructive impact and thus foster creativity, while a competitive approach to conflict, which emphasizes incompatible goals, have a counterproductive impact and will not foster creativity [Barker88].
5.1.3 Participant interpretation enhancement

Participant interpretation enhancement means that all the statements of the language is understood by the participants in the modeling effort.

Underlying basis:

- the phenomena of the language should be easily distinguished from each other
- the number of phenomena should be reasonable
- the use of phenomena should be uniform throughout the whole set of statements possible to express within the language
- the language must be flexible in the level of detail, ie statements must be easily extendible with other statements providing more details, and details must be easily hidden
- separation of concerns, ie it should be possible to divide the models made in the language in natural parts
- expressive economy, ie how many constructs you need to make the statements you want to make.

External representation:

- symbol discrimination should be easy
- the use of symbols should be uniform
- the symbols themselves should be visually simple
- the use of emphasis (ie size, color, font style, etc.) should be in accordance with the relative importance of the statements
- the composition of symbols should be made in a aesthetically pleasing way.

5.1.4 Technical actor interpretation enhancement

Underlying basis
The language should have well-defined semantics and it should lend itself to automatic reasoning.

External representation
All models should have well-defined syntax.

5.2 Domain language requirements

Krogstie’s [Krogstie95] requirements are aimed against languages that are meant to support the later stages of the requirements engineering stage. Our language is meant to support the stage before the requirements engineering stage, ie the domain redesign stage (see chapter 4). Since we are talking about different kinds of languages, we will not fully adopt the language quality requirements developed by Krogstie.
We have fully adopted the requirements in *participant knowledge appropriateness* and *participant interpretation enhancement*. In the two other groups, *domain appropriateness* and *technical actor interpretation enhancement*, we have done some changes.

**Domain appropriateness**
We don’t think it is appropriate to develop a domain language that is powerful enough to express anything in the domain. We don’t think such a language will be practical. We believe the best solution is to have a domain language that is able to express the most important phenomena in the domain and to fill in with natural language or other suitable informal languages where it is appropriate. This view is also advocated by [Karsai95, HP94, Brown90, Davis90, Deville90].

Requirements can be represented using formal languages, semi-formal languages or informal languages. Each of these representational forms has certain strengths and weaknesses, so a combination is necessary. Semi-formal languages are foremost a communication device between users and IS-engineers and offer the necessary freedom at the beginning of the acquisition process [Jarke93]. They might also be used very precisely, at least as precise as formal language [Deville90], but they have not much support for reasoning about the specified requirements. They are therefore unsuitable to support the requirements process in later stages. We will also add that, as for informal languages and semi-formal languages, there are no formal way to control whether a specification based on formal languages is complete or really represents what it is intended to represent.

Domain languages should be used at the beginning of the acquisition process, and one of the objectives of domain languages is to dramatically reduce the communication problems between IS-engineers and users. Thus, domain languages should be semi-formal and not formal.

**Technical actor interpretation enhancement**
Since our domain language should be semi-formal it is not possible to do automatic reasoning with aid of our language.
6 Modeling Languages and Approaches

In this chapter we present the state-of-the-art regarding domain dependent modeling languages, domain independent modeling languages and other user oriented approaches.

6.1 Domain dependent modeling languages

We have found two user-domain oriented language which is meant to be used by users (ie domain experts) to analyze and/or model their own domain: the traditional organization chart modeling language and the operational modeling language used by the military to accomplish operational planning and evaluation. The organization chart language is developed on the behalf of managers who need to know the business’ organizational structure, ie functions and roles, and who is superior to whom. Parts of the military language are presented in chapter 3 Military Symbols for Land Based Systems.

We have found some domain-oriented visual languages. The concepts of visual languages are further described in subsection 6.3.3 Domain-oriented visual programming.

There are a lot of mathematical and technical oriented domain languages, but they are not meant to be used by users (ie domain experts) to analyze their own domain. They are meant to aid the users in solving mathematical and technical problems, eg statistical software, CAD (Computer aided design), etc.

There are also a lot of languages for enterprise/system/software analysis, design and programming, but they are not based on a user-domain’s professional language and they are not developed on the behalf of a specific user-domain. They are all meant to be used by professional analysts, IS-engineers or programmers.

However, we have done some interesting findings regarding domain languages. These findings are presented in the following subsections.

6.1.1 The Florence-project

The objective of the Florence-project [Bjerknes87] was to analyze how nurses could use computer support as an aid in their daily routine work.
In the Florence-project domain languages were assessed. They didn’t call the languages domain languages but *profession oriented programming languages* and *profession oriented system description languages*. The idea behind profession oriented programming languages was that the users (eg nurses) should be able to change the software themselves, while the idea behind profession oriented system description languages was that the system description should be in terms familiar to the users (eg nurses). The objective was to improve the communication between the IS-engineers and the users. Some people in the project didn’t believe in the idea about profession oriented languages and a research was therefore never performed.

The idea behind profession oriented programming languages is the same as the idea behind end user programming as presented in subsection 6.3.3.

The idea behind profession oriented system description languages is the same as our idea behind domain languages. The reason why the Florence-project did no research on profession oriented system description languages was that some people in the project argued that the users would be forced to describe their work on the IS-engineers’ premises. Another argument was that IS-engineers would believe that they had more knowledge about the domain than they really had because profession oriented system descriptions would not give a complete picture of the situation and the problem.

### 6.1.2 The St. Thomas' Hospital project

The objective of the St. Thomas’ Hospital project [Holland94] was to design an information system that would support the business of a clinical directorate.

The St. Thomas’ Hospital project investigated a number of IS modeling languages that could be used to investigate and represent the domain of interest. They rejected all, eg conventional data modeling languages like OOA [Coad91] and those presented in [Hull87], because they tend to focus on information rather than domain behavior and/or because the resulting models have very little predictive power, ie not formal enough, meaning that the models are very difficult to test and validate.

Because of these problems they used a new language for the conduct of the initial domain analysis and subsequent requirements specifications which they termed *Formal Behavioural Analysis*. This language describes the state of the organization and possible behaviors using set theory and the predicate calculus.

The Formal Behavioural Analysis language might be powerful, but set theory and predicate calculus are not user friendly. In [Holland94] it is admitted that it is "virtually impossible to talk to the ‘stakeholders’ using the description - the mathematics has to be translated into English, leading to some of the rigour representation (but not the underlying structure) being lost."

This language is not comprehensible to both users and IS engineers and it does not reduce the communication problems between IS engineers and users. Thus, it is not the kind of domain language we are looking for.
6.1.3 The C2RA - NORCCIS II methodology

C2RA (Command and Control Requirements Analysis) is a NATO defined methodology for analyzing command and control information systems. The C2RA - NORCCIS II\textsuperscript{14} methodology [Flåthen91a] is an extension of the C2RA approach.

The *Command missions* or *Main task* of a HQ are decomposed into *Key processes*, e.g. operations, intelligence, logistics, etc. Key processes are further decomposed into five generic *Key tasks*: status, analysis, plan, decide and execute.

Computer supported Key processes are described in *system concept diagrams*. A general system concept diagram is shown in Figure 6.1 while an example system concept diagram is shown in Figure 6.2.

\textbf{Figure 6.1:} A general system concept diagram [Flåthen91b].

A system concept diagram consists of the following concepts:

- information flow in and out of the headquarters

\textsuperscript{14} NORCCIS II (Norwegian Command, Control and Information System, phase II) is a CCIS project.
- staff elements responsible for executing key tasks
- key tasks
- a description of the computer support containing:
  
  - Generic Application Programs (GAP) which are common for all headquarters
  - Special Application Programs (SAP) which will vary from one HQ to another
  - databases
  - graphical views
  - alphanumeric text views

Every concept is given a unique identifier as shown in Figure 6.2.

The system concept diagrams have two purposes:

- serve as a communication vehicle between users and IS engineers
- document the operational requirements of the computerized parts of the command and control information system.

<table>
<thead>
<tr>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAFF ELEMENT</td>
</tr>
<tr>
<td>.......</td>
</tr>
<tr>
<td>P049N: Land staff</td>
</tr>
<tr>
<td>Presents in-comming info</td>
</tr>
<tr>
<td>PG008 - Own situ. land</td>
</tr>
<tr>
<td>PG003 - Friendly task forces land</td>
</tr>
</tbody>
</table>

Key process: LAND BATTLE MANAGEMENT

Figure 6.2: System concept diagram, example [Flåthen91b].

As can be seen from the two figures, the graphical symbols used in the C2RA - NORCCIS II methodology are mainly based on the traditional flowchart symbols [ISO5807]. The only exception is the symbol used to depict key tasks. However, in
[Flåthen91a] the terminator symbol defined in [ISO5807] is used to depict key tasks. Even if the graphical symbols used in the C2RA - NORCCIS II methodology are based on the traditional flowchart symbols, it seems that the syntax used in the C2RA - NORCCIS II methodology is quite different from the syntax recommended in [ISO5807].

The language used to build system concept diagrams might be classified as a domain oriented language, but it does not completely fit our view on domain languages because it is software oriented and not domain oriented.

6.1.4 The GÖTA method

The foundation of the GÖTA method is user participation. GÖTA models are drawn by the users on walls [Henrikson91]. GÖTA modeling is a kind of enterprise modeling.

The GÖTA method could be divided into three stages: motivation, task description, and modeling.

The objective of the motivation stage is to tell the participants about the purpose of the modeling.

In the second stage every participant writes down, on a piece of paper, in broad terms his/her task (preferably only one noun), what is needed to carry out the task (information, tools, etc), the results of the task, customers, etc. Then the pieces are pasted to the wall(s) and every participant tells the rest of the participants about their task, what is needed to carry out the task, etc.

In the third stage the pieces are grouped and tape-arrows are drawn between related tasks. The relation between tasks is noted on the arrow, preferably only one verb per arrow.

The main objective of GÖTA modeling is not the resulting model, but rather the modeling process. The participants (ie users) get new knowledge about how the enterprise or business domain works, how they fit in, and so on.

The language used in the GÖTA method is simple and therefore easy to use and easy to understand for both users and IS engineers. It might be classified as a domain oriented language, but its expressive power is quite limited.

6.1.5 SAMPO

SAMPO (Speech Act-based information analysis Methodology with comPuter-aided tOols) [Aramäki92] is an office domain oriented method. SAMPO focuses on office communications in terms of speech acts [Searle79] and conversations. The fundamental idea of SAMPO is that "office communications is the key to understand and analyzing offices because it is only through communication (ie discourse) that office workers are able to make sense of their environment". In SAMPO office communications are used as the means to analyze offices. A language for discourse analysis is included in SAMPO.
The SAMPO notation is shown in Figure 6.3.

![Diagram of SAMPO notation]

Figure 6.3: The SAMPO notation.

Even if the SAMPO modeling language might be classified as a domain oriented language, it does not fit our view on domain languages. The SAMPO language is tailored to the office domain, or maybe more correct to the conversations domain, which is a very broad domain and not a specific user-domain. As far as we understand the SAMPO language is meant to be used by professional analysts and IS-engineers and not by the users of the office domain.

### 6.1.6 Object oriented modeling languages

In the domain analysis method presented in PROTEUS [HP94] they recommend to use object oriented modeling languages, e.g. OMT, to build domain models. In the contexts of PROTEUS domain models mean models of domain products/systems formalizing the knowledge on these products, especially modeling their commonalties and variants. In the context of domain modeling as presented in PROTEUS, OMT and other object oriented languages may be appropriate because the PROTEUS domain models are not meant to be a communication vehicle between users and IS-engineers but between IS-engineers in the same company. The PROTEUS domain analysis method could be described as a development strategy for software providers that focuses on evolutionary software development and reuse, and thus reduced development costs.
6.2 Domain independent modeling languages

There are hundreds of different domain independent modeling languages [Siau96]. We have chosen to present a handful of them, three multi-perspective modeling languages and one single-perspective language. The first one is PPP. PPP is a SA/SD (structured analysis and structured design) oriented language. The second one is OMT (Object Modeling Technique). OMT is regarded as a object oriented language. The third one is OOram (Object Oriented role analysis and modeling). OOram is also an object oriented language, but it is different from OMT because the main focus of OOram is on roles and role models and on objects and object models. The fourth, the single-perspective language, is ActionWorkflow. ActionWorkflow is a workflow oriented language.

Non of these languages are what we regard as domain languages because they are not based on a user-domain's professional language and they are not developed on the behalf of a specific user-domain. These languages are developed to be used by IS engineers.

6.2.1 PPP

PPP is an experimental CASE environment that has been developed by the information system group at NTNU, Norway. We will in this subsection give an overview of the PPP language. The overview is based on [Gulla91, Yang93, Lindland93].

PPP consists of four closely interrelated sub-languages:

1. the phenomenon language ONE-R (Our New Entity-Relationship model), which is an extension of the ER language [Chen76]
2. the process language PPM (Process Port Model), which is an extension of the DFD language [Gane79]
3. the process life description language PLD, which is a language for the specification of process logic
4. the actor modeling language AM, which is a language for actor modeling.

ONE-R

ONE-R is claimed to be both an extended ER language and an object oriented language. It is able to describe both real world objects, ie static properties of the domain, and the static objects of the information system.

ONE-R is used to specify the basic phenomena of systems, ie entities and relationships among entities. The concepts entity class and relationship class are used to express this (Figure 6.4). To express properties of the phenomena and possible operations on them, the concepts of data type and method are provided (Figure 6.4).

![Figure 6.4: The basic concepts of ONE-R.](image-url)
An entity class is a set of entities (ie objects) that exist in the domain, while a relationship class is a relation between two or more entity classes. There are two relationship types that may be defined between classes (Figure 6.5):

- **in_a_relationship** exists between an entity class and a relationship class. There are four possibilities (ie cardinals): *full_l, full_n, partial_l* and *partial_n, full* means that any entity in the class must be involved in at least one occurrence of the relationship. *partial* means that the involvement is not mandatory. *l* means involved in one occurrence of the relationship while *n* means involved in one or more occurrences of the relationship. The use of N-ary relationships is similar to ER modeling.

- **is_a_sub_of** is an "including" relationship between two entity classes. It means that all the members of the subclass are included in the super-class. A subclass may also have subclasses, thus an is_a hierarchy can be formed. An entity class may be a subclass of more than one class (multiple inheritance), but it should belong to only one basic class. There are three constraints on the subclasses of an entity class: if an entity class (super-class) has subclasses S1, ... Sn, these subclasses may *make_up* the super-class (= S1 ∪ S2 ... ∪ Sn), or be *distinct* (S1 ∩ S2 ... ∩ Sn = ∅), or both, ie a partition of the super-class.

![Diagram of relationship types](image)

**Figure 6.5: Relationship types.**

Properties of entities and relationships can be expressed by the concept of data type. A data type contains a value set, called a domain. There are four primitive data types defined in ONE-R (Figure 6.6): *integer, real, string[n]* and *boolean*. More complex data types can then be defined by the following constructs (Figure 6.6):

- **renaming**: a new type T is defined by renaming an existing type T'.
- **set**: a new type T is defined on the power set of the value set of type T'. Two numbers may be added to the definition for the minimum and maximum numbers of the elements of a value of type T. The default is (0,n).
- **compositing**: a new type T is a composite type with n components of the types T1, T2, ... Tn. com.1, com.2, ... com.n are the names of the components. Names attached with the symbol "*" indicates an optional component.
- **union**: a new type T is the union of type T1, T2, ... Tn with label lab.1, lab.2, ... lab.n, ie any value of type T is either of type T1, T2 ... or Tn.
In addition to the pre-defined primitive data types, the entity classes may be regarded as abstract data types and thus be used to construct new data types.

The relationship between an entity class or a relationship class and a data type is called an attribute, eg title is one attribute between the entity class paper and the data type STRING (n) (Figure 6.7).

For an entity class or a relationship class, one can also define some methods that define possible operations on the properties on the members of the class. A method is a function which has one or several input values as its parameters and one output value as its result. Two kinds of methods are distinguished according to whether a method updates the value of the attributes of the properties or not. The updating is called a side effect. Methods with side effect are indicated with a filled (gray) method symbol (Figure 6.8).
An example from the domain of field artillery (FA) is shown in Figure 6.9.

![ONER diagram](diagram)

Figure 6.9: ONER, example.

As the small and incomplete example in Figure 6.9 shows, an ONER model is flat and may easily become very large, comprehensive and thus complex. An ONER model should therefore be divided into views, e.g., one or more views for every situation of interest, one or more views showing generalization, one or more showing associations, etc.

**PPM**
The basic concepts of PPM are (Figure 6.10):

- **process**: a specified activity which can transform some input to outputs
- **store**: a place where a collection of data packages is kept
- **timer**: a clock which is used to model events that are to occur at a specific moment in time, or a delay which is used to model events that are delayed with respect to time
- **external agent**: a person, an organization or other things outside the modeled processes which provide inputs to processes, receive outputs from processes, or trigger processes
- **flow**: a movement of data between processes, stores, timers and agents.

![PPM concepts](diagram)

Figure 6.10: Basic concepts.
A flow may have the following properties (Figure 6.11):

- **triggering**: symbolized by attaching a "T" to an input flow of a process. Triggering means that an input can only arrive when the process is not executing (i.e., idle), and that the arrival of the triggering inputs in a legal combination will start the process. At the start of the execution, the process will first receive all the triggering inputs.
- **terminating**: symbolized by attaching a "T" to an output flow of a process. Terminating means that an output can only be sent when the process terminates its execution. All the termination output flows in a legal combination will be sent out before the process changes its state to idle.
- **singular flow**: the process will receive or send only one data package on the flow during execution
- **repeating flow**: the process will receive or send more than one data package on the flow during execution
- **conditional flow**: the process might, but do not need to receive or send a data package on the flow during execution.

![Diagram of flow properties](image)

Figure 6.11: The properties of flows.

To define logical relations between input flows or output flows the concept of *port* is introduced. Three types of ports are defined (Figure 6.12).

- **AND port**: all the member flows of the port are going to be received or sent during an execution of the process
- **XOR port**: one and only one member flow of the port is going to be received or sent during an execution of the process
- **OR port**: at least one member flow of the port is going to be received or sent during an execution of the process.

![Diagram of port types](image)

Figure 6.12: Ports.

The definition of ports is a recursive definition, thus composite ports may be defined
(Figure 6.12).

Normally data flows between components, but possible exceptions are that data packages are thrown away or data packages are lost. To model these cases sinks are defined (Figure 6.13).

![Figure 6.13: Sink.](image)

In the models agents and flows are numbered, eg A1 (agent 1) and f1 (flow 1). Numbers and input/output (I/O) conditions are described on a separate sheet or in a separate window as shown in Table 6.1, Table 6.2 and Table 6.3.

An example from the domain of field artillery (FA) is shown in Figure 6.14, Table 6.1, Table 6.2 and Table 6.3.

![Figure 6.14: PPM, example.](image)

<table>
<thead>
<tr>
<th>Agents and their triggering rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
</tbody>
</table>

[Table 6.1]
Table 6.1: Agents and their triggering rules.

<table>
<thead>
<tr>
<th>Data flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
</tr>
<tr>
<td>f2</td>
</tr>
<tr>
<td>f3</td>
</tr>
<tr>
<td>f4</td>
</tr>
<tr>
<td>f5</td>
</tr>
<tr>
<td>f6</td>
</tr>
<tr>
<td>f7</td>
</tr>
<tr>
<td>f8</td>
</tr>
<tr>
<td>f9</td>
</tr>
<tr>
<td>f10</td>
</tr>
<tr>
<td>f11</td>
</tr>
<tr>
<td>f1.1</td>
</tr>
<tr>
<td>f1.2</td>
</tr>
<tr>
<td>f1.3</td>
</tr>
<tr>
<td>f1.4</td>
</tr>
<tr>
<td>f1.5</td>
</tr>
<tr>
<td>f1.6</td>
</tr>
</tbody>
</table>

Table 6.2: Data flows.

<table>
<thead>
<tr>
<th>P1.1</th>
<th>P1.2</th>
<th>P1.3</th>
<th>P1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2</td>
<td>f2</td>
<td>f5</td>
<td>f9</td>
</tr>
<tr>
<td>f3</td>
<td>f5.3</td>
<td>f1.3</td>
<td></td>
</tr>
<tr>
<td>f4</td>
<td>f6</td>
<td>f8</td>
<td>f10</td>
</tr>
<tr>
<td>f4</td>
<td>X</td>
<td>f7</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>f11</td>
</tr>
</tbody>
</table>

Table 6.3: I/O conditions.

Comments on the I/O conditions of process P1.1: Both f2 and f3 may be received during the process, but only one input is processed at a time, i.e. sequential processing of input.

**PLD**

The PLD language is a software design language. It is used to specify process logic of bottom level processes of PPM models and gives a procedural description of some behavior pattern. A pattern is attachable to both a process or a method, and can involve interaction with other patterns as well as internal computation. Thus, the description of methods can be applied directly from a PLD being part of a process.
The concepts of PLD are (Figure 6.15):

- **start**: indicates the beginning of a PLD diagram
- **receive**: is used to receive data from other PLD models and/or from processes, stores, agents and/or timers of the PPM model. The text inside the receive box specifies the model to receive data from and what data to receive. The receive concept may be succeeded by any concept but the start concept.
- **send**: is used to send data to other PLD models and/or to processes, stores, agents and/or timers of the PPM model. The text inside the send box specifies the model to send data to and what data to send. The send concept may be succeeded by any concept but the start concept.
- **choice**: is used to specify selection, i.e., if and case constructs. The choice concept consists of two sub-concepts: **selection** and **alternative**. The selection concept only indicates a choice and may be succeeded by any concept but the start concept. The alternative concepts are right-connected and each contains a condition. One and only one of the conditions of the alternatives should be true, and that particular alternative will be selected. Alternative concepts are succeeded by any concept but the start concept.
- **assignment**: is used to variable-assignment. It may also be used to symbolize a PLD block or a subroutine call. It succeeds any concepts and may be succeeded by any concept but the start concept.
- **iteration**: is used to specify loop-constructs, i.e., for and while loops. A PLD block must be right connected to the concept. The execution of this PLD block constitutes one iteration. The iteration concept may be succeeded by any concept but the start concept.

![Figure 6.15: PLP concepts.](image)

The PLD syntax is shown in Figure 6.16. The control flow of a PLD model is top-down and left-right.

![Figure 6.16: PLD syntax.](image)
AM
The actor sub-language is used to specify executable programs. The basic concept in AM is the actor. The properties of an actor are represented by functions and a working state as shown in Figure 6.17. A function may receive some inputs and produce some outputs. The same graphical notation is used for functions as for processes. I/O conditions are used to describe the relationships between inputs and outputs of the functions. A working state contains data which describe static aspects of the working state of the actor.

![Diagram](image)

**Figure 6.17**: Example of an actor model.

### 6.2.2 OMT

We will in this subsection give an overview of the OMT (Object Modeling Technique) language. The overview is based on [Rumbaugh91].

OMT is an object oriented method where the principal construct is the object. The object is a discrete, distinguishable entity which encapsulates both the attributes and the operations of the object. Similar objects are grouped together in object classes (classes for short). Attributes and operations are inherited among classes based on a hierarchical relationship.

OMT consists of three orthogonal sub-languages, but it is not clear how the sub-languages and corresponding models are interrelated. The three sub-languages are:

1. the object modeling language
2. the dynamic modeling language
3. the functional modeling language.
The functional model specifies what happens, the dynamic model specifies when it happens, and the object model specifies what it happens to.

**Object modeling language**

This language is used to make object models. An object model describes the static structure of the modeled system by showing the objects in the system, their relationships, and the attributes and operations that characterize each class of objects. Rumbaugh et al. regards the object modeling language as the most important of the three languages because they recommend building a system around objects rather than around functionality. Their arguments are that object models more closely correspond to the real world and that object models are more resilient with respect to change.

The object modeling language is very comprehensive as shown below.

A class is represented by a rectangle, and the rectangle may be filled with the class’ attributes and operations (Figure 6.18).

![Class diagram](image)

**Figure 6.18: Class.**

An object is represented by a rectangle with rounded corners. Attribute values may be explicitly shown in the symbol. The relationship between a class and its objects is represented by a dashed arrow (Figure 6.19).

![Object instance diagram](image)

**Figure 6.19: Object instances and instantiation relationship.**

Links and associations are the means for establishing relationships among objects and classes respectively. The supported associations are shown in Figure 6.20.
Figure 6.20: Associations.

Note:

1. A qualifier is a special attribute that reduces the effective multiplicity of an association. It is applied to the many side of an association.
2. Ordering is used to indicate that objects on the many side of an association are ordered.
3. A link attribute is a property of the links in an association.
4. Aggregation uses the multiplicity symbols to give the number of components.

Generalization shows the inheritance hierarchy (Figure 6.21).
Note:

1. A discriminator is an attribute whose value differentiates between subclasses.
2. Black triangle denotes that the subclasses have overlapping (nondisjoint) membership.

Other modeling concepts are: abstract operation, class attributes and class operations, propagation of operations, constraints on objects, derived attribute and derived class.

An example from the domain of field artillery (FA) is shown in Figure 6.22.

As our small and incomplete example in Figure 6.22 shows, an object model is flat and may easily become very large, comprehensive and thus complex. An object model should therefore be divided into views, e.g., one or more views for every situation of
interest, one or more views showing generalization (inheritance), one or more showing aggregation, one or more showing associations, etc.

**Dynamic modeling language**
A dynamic model consists of multiple state diagrams, one state diagram for each class with important dynamic behavior. The dynamic model shows the pattern of activity for the entire system. Each state diagram shows the behavior of an object class as seen from its inside, and each state machine, visualized by a state diagram, executes concurrently and may change state independently. The state diagrams are glued together by shared events.

The major dynamic modeling concepts are *events*, which represent external stimuli, and *states*, which represent values of objects (Figure 6.23). When an event is received, the next state depends on the current state as well as the event. A change of state caused by an event is called a *transition*.

![Figure 6.23: States and a simple event.](image)

An *activity*, which is an operation that takes time to complete, is associated with a state. The notation "do: activity-A" within a state box indicates that activity-A starts on entry to the state and drops on exit (Figure 6.24). An *action*, which is an instantaneous operation seeming to take no time to complete compared with activities, may be associated with the entry to ("entry-action") and/or exit from ("exit-action") a state. An action may also be associated with an event that does not cause a transition from the state ("event1 / action1") (Figure 6.24).

![Figure 6.24: State box.](image)

Events that result in a transaction may be followed by one or more attributes within parentheses, a guard, which consists of conditions, within square brackets, and/or an action (Figure 6.25).

![Figure 6.25: Event and optional constructs.](image)
Events may also be sent to object classes outside the modeled object class as shown in Figure 6.26.

![State diagram](image)

**Figure 6.26:** Event sent to an object class.

One-shot state diagrams represent objects with finite life. One-shot diagrams have initial and final states which both might be labeled (Figure 6.27). The initial state, shown by a solid circle, is entered on creation of an object. A final state is represented by a bull’s-eye.

![State diagram](image)

**Figure 6.27:** Initial and final state.

To avoid flat state diagrams, which soon will become impractical even for quite small problems, and to extend the expressive power, state generalization (nesting) and concurrency are introduced (Figure 6.28)\(^\text{15}\).

![State diagram](image)

**Figure 6.28:** State generalization and concurrent sub-diagrams.

Usually an object in a superstate must be in exactly one of the sub-states, but a state may be divided into concurrent sub-states. Concurrent sub-states are distinguished by a dashed line. Concurrent states can be synchronized as shown in Figure 6.29.

---

\(^{15}\) Based on the notation of David Harel [Harel87]
Scenarios, which are a textual listing of a sample sequence of events that may occur during an execution of the system, and event traces, which are quite similar to the scenario view in OOram (see subsection 6.2.3), may be used to aid the construction of state diagrams.

**Functional modeling language**
The functional model describes the computations within a system. The functional model should specify the meaning of the operations in the object model and the actions in the dynamic model.

The notation used is similar to traditional DFD (Figure 6.30).

![Diagram](image)

**Figure 6.30:** The notation similar to traditional DFD.
As shown in Figure 6.30, the external entity of the original DFD is replaced by an actor object. An actor is an active object which drives the data flow by producing or consuming data. An actor is drawn as a rectangle to show that it is an object.

The functional language offers mainly three extensions compared to the traditional DFD:

1. the possibility of sending control flows (which are signals only) between processes (Figure 6.31)

```
process-1 -------- boolean result -------- process-2
```

Figure 6.31: Control flow.

2. the possibility of dynamically creating data stores (Figure 6.32). Data stores correspond to objects and the creation of a data store imply the instantiation of an object

```
Name of data store
```

Figure 6.32: Data flow that results in a data store.

3. the possibility of composing, decomposing and duplicating data values, i.e., data flows (Figure 6.33).

```
Composition
```
```
Decomposition
```
```
Duplication
```

Figure 6.33: Data flow.

An example from the domain of field artillery (FA) is shown in Figure 6.34.
6.2.3 OOram

We will in this subsection give an overview of the OOram (Object Oriented role analysis and modeling) language. The overview is based on [Reenskaug96].

Like OMT, OOram is an object oriented method. However, the main focus of OOram is on roles and role models and not on objects and object models. Roles are a further abstraction of objects, but they have all the properties of objects. Roles have identity and are encapsulated, and as such roles are defined by their attributes and the messages they may send and receive.

"A role model is part of a structure of objects which we choose to regard as a whole, separated from the rest of the structure during some period of consideration. It is a whole that we choose to consider as a collection of roles,
with each role being characterized by its attributes and by the messages it may receive from and send to other roles" [Reenskaug96].

OOram combines the expressiveness of the concepts of object and class which are found in most object oriented methods. All information that can be expressed in a class model can be expressed in a role model, and all information that can be expressed in an object model can be expressed in the same role model. In simple cases, the role model is identical to the object model and there is exactly one role for each object. The notion of a class focuses on the capabilities of the objects while the notion of a role focuses on the position and responsibilities of an object within the overall structure of objects. According to [Reenskaug96], the role model abstraction belongs to the realm of modeling while the class model abstraction belongs to the realm of implementation. Thus, a role model constitutes an abstracted runtime-description for an object collaboration structure, while a class model constitutes a build-description as object templates.

An object can play several roles. This permits a systematic separation of concerns by describing phenomena using different role models. And conversely, it permits the synthesis of a derived role model by letting its objects play several roles from different role models.

The essence of the object oriented paradigm is the modeling of interesting phenomena as a structure of interacting objects. In an OOram role model, patterns of interacting objects are abstracted into a corresponding pattern of interacting roles. Role models support a very general separation of concerns, and the notion of role model synthesis supports the construction of complex role models from simpler ones in a safe and controlled manner.

The system of interacting roles may be observed from different observation points. OOram supports three different points of observation, called perspectives:

1. the environment perspective, where the observer is placed in the system's environment so that she can observe the system's interaction with its environment roles (Figure 6.35).
2. the external perspective, where the observer is placed between the roles so that she can observe the messages flowing between them (Figure 6.36).
3. the internal perspective, where the observer is placed inside a role so that she can observe its implementation (Figure 6.37).

Figure 6.35: The environment perspective.
The system of interacting roles may be studied in different views. Each view expressing certain aspects of the system of roles while suppressing others. OOram supports ten different views. The views are different presentations of the same role model. They are not orthogonal, i.e., their mutual consistency could be enforced automatically. The ten views are:

1. the area of concern view
2. the stimulus-response view
3. the collaboration view
4. the scenario view
5. the interface view
6. the semantic view
7. the process view
8. the state diagram view
9. the method specification view
10. the role list view.

Each view is expressed by a language.
The views are only meaningful in certain perspectives, as shown in Table 6.4.

<table>
<thead>
<tr>
<th>View</th>
<th>Environment</th>
<th>External</th>
<th>Internal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area of concern view</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stimulus-response view</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaboration view</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Scenario view</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Interface view</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Semantic view</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Process view</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>State diagram view</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Method specification view</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Role list view</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Usefulness of the views in different perspectives.

**Area of concern language**
The area of concern view is a textual description of the phenomenon modeled by the role model. Thus the used language is a natural language like Norwegian or English (Figure 6.38).

![This role model describes the fire planning process executed in the field artillery.](image_url)

Figure 6.38: Are of concern view, example.

**Stimulus-response language**
The stimulus-response view shows:

- the stimulus message sent from an environment role to the system
- response messages sent from the system to one or more environment roles, or some other changes that are described as a free text, eg changes to object structure or attributes.

The stimulus message is a trigger that triggers the system, ie a trigger cause some activities to happen in the system. The effect of the activities on the environment are called response messages.

The stimulus-response language is structured natural language. Its syntax is shown in Table 6.5 and an example is shown in Table 6.6.

<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response messages</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment role</td>
<td>Environment role</td>
<td>Free text description of</td>
</tr>
<tr>
<td>&gt;&gt; message name</td>
<td>&lt;= message name</td>
<td>other results</td>
</tr>
</tbody>
</table>

Table 6.5: Stimulus-response language.
<table>
<thead>
<tr>
<th>Stimulus message</th>
<th>Response messages</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSCC/Brigade</td>
<td>FSCC/Division</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt; Art.ops</td>
<td>&lt;&lt; Fire plan ready</td>
<td></td>
</tr>
<tr>
<td>FSCC/Brigade</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;&lt; Fire plan ready</td>
<td></td>
</tr>
<tr>
<td>Battery commander</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;&lt; Fire plan ready</td>
<td></td>
</tr>
<tr>
<td></td>
<td>There are three battery commanders.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.6: Stimulus-response view, fire planning.

**Collaboration language**

The collaboration view shows the roles and the message paths between them. The language concepts are shown in Table 6.7 and an example is shown in Figure 6.39.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super-ellipse</td>
<td>a system role.</td>
</tr>
<tr>
<td>Dashed super-ellipse</td>
<td>an environment role.</td>
</tr>
<tr>
<td>Divided super-ellipse</td>
<td>a role where attribute names are listed.</td>
</tr>
<tr>
<td>Super-ellipse with a shadow</td>
<td>a virtual role. A virtual role is a role that represents a cluster of objects rather than a single object. Virtual roles are artifacts of the presentation and do not exist in the underlying role model.</td>
</tr>
<tr>
<td>Line</td>
<td>a message path between collaborating roles.</td>
</tr>
<tr>
<td>Small circle</td>
<td>a simple port, indicating that the adjacent role knows about exactly one collaborator role.</td>
</tr>
<tr>
<td>Small double circle</td>
<td>a multiple port, indicating that the adjacent role knows about any number of collaborator roles, one of which is shown in the view.</td>
</tr>
<tr>
<td>Cross</td>
<td>no port, indicating that the adjacent role does not send messages to the collaborator role.</td>
</tr>
<tr>
<td>Dashed open rectangle</td>
<td>comments or explanations.</td>
</tr>
</tbody>
</table>

Table 6.7: Collaboration language.
The developers of OOram have noticed the value of using graphical symbols (i.e., decorated roles) to indicate the nature of the roles. Role symbols may therefore be decorated with graphical symbols. The IS engineers are free to define their own symbols, e.g., military symbols as shown in Figure 6.40, but note that these symbols will not become an integrated part of the language and thus will not be included in consistency controls and synthesis operations.

![Figure 6.39: Collaboration view, fire planning 1.](image)

![Figure 6.40: Collaboration view, fire planning 2.](image)

**Scenario language**
The scenario view shows a specific, time-ordered sequence of interactions between roles. The scenario view shows a sample message sequence, thus several scenario views
may be connected to a single role model, each view showing a specific, time ordered sequence of message exchanges. The first message must be a stimulus message.

The scenario language consists of the same type of roles as the collaboration language. In addition it consists of a time line (vertical line connected to the roles) and six types of interaction (Table 6.8).

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unspecified interaction</td>
</tr>
<tr>
<td>S</td>
<td>synchronous interaction</td>
</tr>
<tr>
<td>D</td>
<td>synchronous deferred interaction</td>
</tr>
<tr>
<td>A</td>
<td>asynchronous interaction</td>
</tr>
<tr>
<td>C</td>
<td>creation of receiver prior to interaction</td>
</tr>
<tr>
<td></td>
<td>method return</td>
</tr>
</tbody>
</table>

Table 6.8: Types of interaction.

An example is shown in Figure 6.41. Note that some messages are sent to more than on receiver. The OOram language have no concept for modeling such situations, thus we have modeled this as a sequence of equal messages being sent from the sender role.
Figure 6.41: Scenario view, fire planning.
Interface language
The set of messages that may be sent from one role to another are shown in the interface view. Interfaces are usually specified textually (Figure 6.42), but may also be specified in rounded rectangles (Figure 6.43) connected to the ports in the collaboration view (Figure 6.44).

The interface language is structured natural language. A rounded rectangle with a dashed line should be used to denote an interface in the collaboration view. The dashed line should be connected to the port described by the interface. The interface name is read as follows: To FA Battalion from FSCC/Brigade (see Figure 6.42 and Figure 6.44).

```plaintext
interface 'FA Battalion<FSCC/Brigade'
  message synch 'Art.ops'
    explanation "Give artillery operation order"
interface 'FSCC/Brigade<FA Battalion'
  message synch 'Fire plan'
    explanation "Send fire plan to FSCC/Brigade if the number of maneuver battalions involved are greater than one"
  message synch 'Fire plan ready'
    explanation "Send fire plan ready to FSCC/Brigade when at least two FA Batteries are ready"
interface 'FA Battery<FA Battalion'
  message synch 'Fire plan data'
    explanation "When received fire plan data from a battery, circulate these to the other batteries"
  message ..........  
```

Figure 6.42: Interface view, textual example.

---

<table>
<thead>
<tr>
<th>Interface name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message list</td>
</tr>
</tbody>
</table>

Figure 6.43: Interface language, graphical notation.
As can be seen from Figure 6.44 the graphical form should only be used in very simple cases.

**Semantic language**
The semantic view describes the meaning we associate with the roles and their relationships (Figure 6.45). According to [Reenskaug96] the semantic view is rarely needed because the collaboration view usually contains sufficient information.

The semantic language consists of the same type of roles as the collaboration language, but the instead of message links the semantic language supports the concept of relation and cardinality (Table 6.9).

<table>
<thead>
<tr>
<th>Relation</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>relation</td>
<td>exactly one</td>
</tr>
<tr>
<td></td>
<td>zero or one</td>
</tr>
<tr>
<td></td>
<td>one or more</td>
</tr>
<tr>
<td></td>
<td>zero or more</td>
</tr>
</tbody>
</table>

Table 6.9: Relations and cardinalities.
Figure 6.45: Semantic view, fire planning.

**Process language**
The process view shows the flow of data between the roles, and the processing of the data in the roles. According to [Reenskaug96] the process language is especially useful for modeling the flow of data and work procedures in human organizations.

The process language consists of the same type of roles as the collaboration language. In addition the symbols shown in Table 6.10 are included in the process language.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelogram</td>
<td>a data set which is transferred as message parameters.</td>
</tr>
<tr>
<td>Rectangle</td>
<td>an action performed on received data.</td>
</tr>
<tr>
<td>Arrow</td>
<td>data transmission in direction of the arrow.</td>
</tr>
</tbody>
</table>

Table 6.10: Process language.

The process view of the fire plan management process executed in the artillery battalion headquarters is shown in Figure 6.46.
Figure 6.46: Process view, fire plan management.

State diagram language
The state diagram view shows the legal states of a role and the messages that trigger a transition from one state to another.

The state diagram language consists of four modeling concepts as shown in Table 6.11.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Refers to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large circle</td>
<td>state.</td>
</tr>
<tr>
<td>Line</td>
<td>transition. The line may be annotated with the message names.</td>
</tr>
<tr>
<td>Small circle</td>
<td>one or more messages.</td>
</tr>
<tr>
<td>Small dashed circle</td>
<td>no messages.</td>
</tr>
</tbody>
</table>

Table 6.11: State diagram language.

A simplified state diagram view of the firing process executed by the gun groups is shown in Figure 6.47.

![State diagram view, firing]

Figure 6.47: State diagram view, firing.

**Method specification language**

The method specification view is similar to the scenario view. The main difference is in the perspective. The scenario view shows a specific sequence of message transmissions from the environment perspective or the external perspective. The method specification view shows the general processing (taking care of all alternative actions) of a message from within a specific role. It shows the message reception, the triggered method\textsuperscript{16}, and the message sent from that method (Figure 6.48).

The method specification language is similar to the scenario language, but in addition it has a rectangle where the method is named.

\textsuperscript{16} Note that in OMT a method is the implementation of an operation. Thus, method in OOram corresponds to operation in OMT.
Role list language
The role list view gives an overview of the roles, their names, purpose, and attributes.

The used language is structured natural language. Its syntax is shown in Figure 6.49.

```
role 'name of role'
explanation "textual explanation"
attribute 'name of attribute'
   explanation "textual explanation"
role ............
```

Figure 6.49: Role list view, syntax.

Synthesis
"We always seem to extend the scope of our systems, and even object models are frequently too complex to be comprehended by our limited brain capacity. The OOram technology provides abstractions that help us to divide and conquer, enabling us to handle complex phenomenon to create simple role models, and understand each of these models separately." [Reeskaug96]

The term synthesis is in OOram used to denote the construction of complex models from simpler ones.

6.2.4 ActionWorkflow®

We will in this subsection give an overview of the ActionWorkflow language. The overview is based on [Medina-Mora92 and Action93].

ActionWorkflow has its foundation in the speech act theory and focuses on modeling workflow as a co-ordination among people.

In an ActionWorkflow loop (Figure 6.50) there should always be an identified customer and an identified performer where the performer agrees to complete a particular action to the satisfaction of the customer. The ultimate goal of the loop is customer satisfaction. Thus, the loop has to be closed.
Figure 6.50: The concepts of Action Workflow.

The Action Workflow loop has four phases:

1. *proposal*: the customer makes a request for work, or the performer makes an offer to the customer
2. *agreement*: the customer and the performer reaches a mutual agreement on the conditions of satisfaction
3. *performance*: the performer performs what has been agreed upon and declares to the customer that the action is completed
4. *satisfaction*: the customer assess the work according to the conditions of satisfaction and declares satisfaction or dissatisfaction.

The main workflow, which describes the principal customer's condition of satisfaction, is called primary workflow (Figure 6.51). The other workflows, which describes the coordination between people required to meet the conditions of satisfaction, are called secondary workflows (Figure 6.51).

Figure 6.51: Types of links.

There are four link types between workflows:

- conditional or unconditional.
- normal flow or exception flow
parallel or serial
act-driven or state-driven.

The first three types of links are shown in Figure 6.51. There is no graphical notation for illustrating act-driven or state-driven links. They are described through specific link definition dialogs. Conditional links are shown as diamond decision boxes in the middle of the line. All links without this diamond are unconditional links. Normal flow links are shown by solid lines, while exception flow links are shown by dashed lines.

An example from the domain of field artillery (FA) is shown in Figure 6.52.

![Figure 6.52: ActionWorkflow, example.](image)

### 6.3 Other user oriented approaches

In this section we present other user oriented approaches, ie approaches focusing on user involvement, communication between users and developers, and domain knowledge acquisition.
6.3.1 Joint application development

The first concept of joint application development (JAD) originated at IBM. The first JAD sessions were held at IBM's offices in Raleigh, North Carolina, in 1977. Originally, the "D" in the acronym JAD stood for "Design". Currently, there are differing opinions about whether the "D" means "Development" or "Design" [Wood95].

JAD centers on a three- to five-day structured workshop session that brings together users affected by the information system and IS professionals. Under the direction of a facilitator, these people decide on anything from high-level strategic plans to detailed system specifications. The products of the JAD session can include definitions of business processes, prototypes, requirements specifications, data models, etc [Wood95]. According to [Wood95] the advantages of using JAD includes a dramatic shortening of the time it takes to complete a project because of commitment, group cohesion, productive meetings, good communication, common views and understanding. JAD is also claimed to improve the quality of the final product by focusing on the first phases of the development life cycle, thus reducing the likelihood of errors that are expensive to correct later on.

The five phases of JAD are [Wood95]:

1. JAD project definition
2. Research, ie gathering more details about the system and the user requirements
3. Preparation of the workshop session
4. The workshop session
5. Production of the final document.

Traditionally, JAD has been a joint venture between users and IS engineers. In recent years, it has also become a joint venture among any people who need to make decisions affecting multiple areas of an organization.

Related approaches, but not restricted to, are:

- **participatory design (PD)** "The field of participatory design spans a rich diversity of theories, practices, analyses, and actions, with the goal of working directly with users (and other stakeholders) in the design of social systems including computer systems that are part of human work." [Kuhn93]. PD is often termed the "Scandinavian way" while JAD is often termed the "American way". PD advocates a much stronger form of user involvement than JAD. PD is also more creative than JAD but less structured [Carmel93].

- **cooperative design** [Greenbaum91, Kyng95].

- **rapid application development (RAD)** [Martin91]. JAD is the fundamental methodological basis for Martin's RAD [Carmel93].

- **user-centric software engineering (USE)** "The aim of USE is to facilitate collaboration between users and developers to better meet user needs. The result is increased software productivity and quality." [DeBellis95].

- **customer-centered design** [Holtzblatt93]. This approach is more context focused than the other approaches, ie all design work should be done in the users own
context. "Use the users well. Let their own context strengthen them." [Holtzblatt93].

- **Contextual Inquiry** is a field requirements gathering technique used by design teams to get the detailed perspective they need on the customer. An apprenticeship model based on the traditional relationship between master and apprentice is the fundament idea of Contextual Inquiry (where the customer is the master and the designer is the apprentice). "The apprenticeship model provides an attitude of inquiry and learning to adopt while in the field. It recognizes that the customers are the experts in their work and the designers are not" [Beyer95]. Contextual Inquiry is closely related to the previous approach: customer-centered design.

The concept of JAD, PD, etc correspond to our view of user involvement. We have emphasized user involvement in the introduction to this thesis, and the idea of user involvement is our motivation for this thesis. We have developed a domain oriented modeling language that should be used to obtain good communication between users and IS engineers and thus stimulate to user involvement. A domain modeling language should be part of the toolbox used by concepts like JAD and PD.

"Let modeling languages help you."[Holtzblatt93]

### 6.3.2 Prototyping

The main idea behind prototyping is to build a model, ie prototype, of the software system that can undergo trials with the users of the software system. This should help the users to define their requirements more precisely and allow the IS engineers, by evolution of the prototype, to produce a model of the required system behavior and performance against which to develop the final system [Johnson96].

Prototypes are working models of a software system. IS engineers work interactively with the users to build prototypes before constructing the real system. Prototyping makes it possible to turn user requirements into working models of a software system. Thus, it is possible to see the design early in the life cycle and to modify it if deemed necessary. In the end, the prototype more accurately reflects the user requirements than if the system were designed from hardcopy specifications. Prototyping is a trial-and-error approach [Wood95].

There is little doubt that prototyping has its merits, but it also has its drawbacks. Be aware of [Wood95]:

- it is tempting to cut corners on systems analysis and design
- prototypes are not finished systems, prototypes are models
- know when to stop. Don’t fall into a never ending cycle. Prototypes encourage users to change their minds. This is good, up to a certain point.

There are at least three approaches to prototyping:

1. **throwaway prototyping** [Gomaa81] is an informal development approach.

   Throwaway prototypes are constructed in a quick-and-dirty manner with a minimum
of documentation and quality assurance [Davis90].

2. **evolutionary prototyping** [Balzer82] is a more formal development approach. Evolutionary prototypes are constructed in a quality manner with few if any shortcuts. The main intention behind this approach is to build prototypes that evolve into finished software products [Davis90].

3. **mock-up prototyping** [Vonk90] is an informal development approach. Mock-up prototypes are also constructed in a quick-and-dirty manner with a minimum of documentation and quality assurance. The main intention behind mock-ups are to build prototypes of the MMI (man-machine interface), i.e. the external appearance of the software system and the human-computer interaction.

The main difference between prototyping and domain-oriented modeling is that the former focuses on software implementation and software oriented challenges while the latter focuses on domain understanding and domain oriented challenges. Thus, in connection with information systems development domain modeling is normally initiated before prototyping (see chapter 4). However, they are both aiming at the same goal: understanding the users’ needs. Prototyping and domain modeling should both be used in frameworks like participatory design and JAD.

### 6.3.3 Domain-oriented visual programming

The idea behind domain-oriented visual programming, also called end-user programming or end-user computing, is to enable the domain experts, i.e. users, to directly develop and modify software systems. Domain-oriented visual programming languages should describe a program through graphical or spatial expressions rather than through character sequences as in conventional general purpose programming languages [Mehandjiev96].

Domain-oriented visual programming languages must exploit the fact that domain experts will be solving problems that are specific to their particular domain and that they have no interest in learning general purpose programming languages. Domain-oriented visual programming languages should therefore use visual expressions and terminologies that reflect the needs and vocabulary specific to a domain, i.e. they should provide task-specific programming constructs, constructs that represent concepts from the domain. The graphical dimension of visual programming languages can be exploited to present these concepts effectively [Mehandjiev96].

The inherent weakness of visual programming languages is the scaling-up problem [Burnett95]. They are unsuitable for large, integrated and complex software systems.

"The directness, immediacy, and simplicity of visual programming languages are appealing. The question is, can visual programming languages be effectively applied to large-scale programming problems while retaining these characteristics?" [Burnett95]

There is a great deal of activity today in the area of domain-oriented visual programming and visual programming environments in general, both in research and in
practice.

The main difference between domain-oriented visual programming and domain-oriented modeling is that the former focuses on software implementation and software oriented challenges while the latter focuses on domain understanding and domain oriented challenges.
7 The Domain Language

7.1 Introduction

The objective of our domain language is to provide a language that makes it easier to analyze and understand the domain of military CCI systems than it is today, and to document the domain in a more structured way than simply providing natural language descriptions.

It is important to use our domain language in a group oriented modeling approach. A group oriented modeling approach where users and IS engineers are modeling together as a team is the key to proper communication and common understanding between the two.

As mentioned in chapter 2 our domain language should be tailored to the command and control part of the CCIS, i.e. the domain language should be able to express the most important aspects of C&CIS. In our view the most important characteristics of C&CIS could be summarized as follows:

- narrow decision window
- information intensive
- low transmission rate.

From these characteristics we are able to derive the most important aspects of C&CIS:

- a narrow decision window motivates efficient and effective processes, clear cut responsibilities (roles), relevant information, IT support and a goal driven organization
- information intensive systems motivate available and relevant information and thus information processing capabilities
- low transmission rate motivates strict information passing, i.e. message passing, between units.

In addition to the these aspects (process, information, role, IT and goal) it is important to be able to describe the organization a C&CIS is part of and thus influenced by.
These six aspects are depicted in Figure 7.1.

Figure 7.1: Aspects.

We apply six sub-languages and four information tables in order to visualize these six aspects. The six aspects are highly interdependent. A change in any one will usually results in compensatory changes in others.

The six sub-languages are:

- the process overview sub-language
- the process sub-language
- the requirement, goal, problem and mean sub-language (RGPM sub-language)
- the message sequence sub-language
- the role sub-language
- the organization structure sub-language.

Note that we have no separate IT modeling sub-language. The IT modeling sub-language is part of the process sub-language and the message sequence sub-language.

The four information tables are:

- the message content table
- the responsibility table
- the rule table
- the goal table.

The sub-languages and tables are closely interrelated.

The subsequent sections present our domain language, ie the sub-languages and the tables. Note that examples presented in this chapter don't describe the Norwegian artillery as it is or is planned to be. To avoid classified information we have done some modifications to its structures, working process, and so on.
### 7.2 Metamodel

We have used the referent language to make a metamodel of our domain language. The referent language is based on a set theoretic approach, and it is developed by the information system group at NTNU [Østvoll95, SØlvberg97].

The metamodel notation is presented in Figure 7.2, Figure 7.3 and Figure 7.4. The basic symbols are shown in Figure 7.2.

- **Referent**: A referent is a set of objects in the real world which share the same type of attributes and relations with other objects.

- **Attributes**: An attribute collection consists of a set of attributes.

- **Element**: An element describes a single unique element in a set.

- **Relation**: A relation is used for recursive relations or n-ary relations. It can be interpreted as a set.

- **Datstore**: A datastore gives a symbol representation of a corresponding referent

![Figure 7.2: Basic symbols.](image)

In Figure 7.3 constraints included in the referent language are shown. Constraints are used to indicate different types of abstraction mechanisms. Specialization arrows are used together with *member-of*, *subset-of* and *partition-of* constraints.

![Figure 7.3: Constraints.](image)

The relations used between referents, and the relations used together with the *part-of* constraint are shown in Figure 7.4. A black dot (see Figure 7.4) is used to indicate full coverage of a relation.
Relations and cardinalities

\[ \begin{align*}
\text{name} & : 1 : m \\
\text{name} & : 1 : 1 \\
\text{name} & : n : m
\end{align*} \]

Part-of cardinalities

\[ \begin{align*}
\text{1 : m} & \quad \text{1 : 1} \quad \text{n : m}
\end{align*} \]

- Full coverage

Figure 7.4: Relations and cardinalities.

A small example referent model is shown in Figure 7.5.

![Diagram of a referent model with entities and relationships](image)

Figure 7.5: Example referent model.

In Figure 7.6 the top level of the metamodel of our domain language is depicted. The metamodel view of the different sub-languages is presented in the subsequent sections. A complete metamodel showing the integration of the different sub-languages into a compound language is presented in section 7.9.

![Diagram of the metamodel](image)

Figure 7.6: The top level of the metamodel.
In Figure 7.7 the concept of unit is shown. In the sequel it is important to be aware of the concept of unit and its partitions.

![Diagram of unit and related components]

Figure 7.7: Unit.

### 7.3 The process overview sub-language

#### 7.3.1 Introduction

Process overview models, created by the process overview sub-language, give us an overview of the main processes, i.e., customer processes, support processes and rule processes, in the domain.

Customer processes are processes that serve the customers of the domain while support processes are processes that serve customer processes (i.e., customer processes are customers of support processes) or other support processes. Rule processes rule other processes.

Normally, it is maneuver units like armor and infantry (Figure 7.8) that are the customers of the field artillery, but every unit, including the artillery itself, may of course call for fire.
7.3.2 The process overview language

The process overview sub-language has two modeling concepts:

- process
- unit

Processes shown in a process overview model are the main processes in the domain. The process symbol in this language is similar to the process symbol in the process language, but the syntax of a process description is different because the same expressive power is not needed. The syntax of a process overview description has the following characteristics (Figure 7.9):

- size indicator showing at which level the process is executed (optional)
- process name gives a short description of what the process is doing
- P no, process number, showing a unique process identifier
- Unit showing in which type of unit the process is executed.

It is mandatory to use P no, unit and process name.

Units are customers of the customer processes.

Processes and units are connected by relations. There are three relations as shown in Figure 7.10 and Figure 7.12: rules, supports and serves.

7.3.3 Metamodel

The metamodel of the process overview sub-language is depicted in Figure 7.10.
Note, an overview process is owned by a unit which serves other units.

7.3.4 Method

The process overview modeling method has four steps:

1. find those processes that directly support the customers of the domain, ie customer processes
2. find those processes that support the customer processes, ie support processes
3. find processes that support other support processes
4. find processes that rule other processes, ie rule processes.

If the number of customer processes in a domain are higher than one it is reasonable to believe that there is a redesign potential or that the domain boundary are inappropriate.

If the number of support processes per customer process are higher than five or the level of support processes are higher than one (ie, there are processes that support support-processes) it is reasonable to believe that there is a redesign potential.

If the number of rule processes in a domain are higher than one it is reasonable to believe that there is a redesign potential.

The given figures for when to believe it is reasonable to do redesign are not based on research. It is our own guidelines. There are of course other indications too for when to
7.3.5 Example

Customers of the artillery associate a big "bang" with artillery (Figure 7.11 a)), and the big "bang" is the result of the firing process (Figure 7.11 b)), ie the firing process is the customer process.

![Diagram of the customer process in the domain of artillery](image)

Figure 7.11: The customer process in the domain of artillery

![Diagram of a process overview model for the artillery](image)

Figure 7.12: Process overview model for the artillery.
Figure 7.12 shows that there are two customer processes, the firing process, which typically serves armor, infantry, artillery, train and engineer units, and the intelligence process, which serves the brigade. Four processes support the firing process: maneuvering, fire planning, logistics, and establish technical baseline. In addition Figure 7.12 shows that the customer processes and the support processes are ruled by orders made by the rule process: make orders.

7.4 The requirement, goal, problem and mean sub-language

7.4.1 Introduction

In the sequel the requirement, goal, problem and mean sub-language will be called the RGPM sub-language.

We should have high level requirements and goals for all the main processes, ie customer processes, support processes and rule processes, in the domain.

It is very important to know the goals of the main processes of a domain before we start to model it. We can neither investigate if the domain does the right things to the right time nor model how it should work if we don't know the goals of its main processes.

It is also of interest to show problems that might hamper our goals and possible means to overcome these problems. This will give us the first overview of possible solutions to our problems.

7.4.2 The RGPM language

The RGPM language has six modeling concepts:

- customer
- requirement
- goal
- problem
- mean
- achiever.

It is mandatory to use customer, requirement, goal and achiever.

Customers. For every process there should be at least one customer. In the RGPM language we are concentrating on the main processes and thus the customers of the main processes.
Customer of the <process> process

Figure 7.13: Customer.

The customer concept has the following characteristics (Figure 7.13):

- a symbol representing the customers of the process. If the process is a customer process the symbol will be a unit, an observation post or an installation (Figure 7.14 a)). If the process is a support process the symbol will be a process (Figure 7.14 b))

![Customer of the firing process](image)

![Customer of the fire planning process](image)

a)  

b)  

Figure 7.14: Example of customers.

- an underlined text string specifying the main process (eg as shown in Figure 7.14).

Requirements, goals, problems and means are written in plain text in rectangles. Note, only one requirement and one goal per rectangle.

Achievers, ie units, headquarters and equipment are represented by military symbols as defined in chapter 3. Units and headquarters are easily distinguished because field G (se subsection 3.2.3) is attached to headquarters to show the particular center of interest (see Figure 7.15).

Requirements affect achievers, and goals are assigned to achievers. If a requirement affects more than one achiever the military symbols are put into a rectangle, eg Figure 7.15.

![Achievers: Fire control center (FCC) and gun group](image)

Figure 7.15: Achievers: Fire control center (FCC) and gun group.

The syntax of the RGPM language is shown in Figure 7.16. Note that a goal might be divided into sub-goals.

Figure 7.16 should be read as follows (left branch): Customers of the specified process have "requirement 1", "requirement 2", etc. "requirement 1" affects "achievers 1" and "achievers 2". To be able to fulfill "requirement 1", "achievers 1" has been assigned
"goal 1". "goal 1" might be hampered by "Problems 1", but these problems might be overcome by "Means 1". Note that every problem and every mean begins with a boldfaced letter.

To facilitate the readability of RGPM models, we strongly recommend to view or present RGPM models requirement by requirement as shown in Figure 7.20.

![Diagram of RGPM model]

Figure 7.16: Syntax.

### 7.4.3 Metamodel

The metamodel of the RGPM sub-language is depicted in Figure 7.17.
7.4.4 Method

The RGPM modeling method has the following steps:

1. for every main process list the major contributors, ie HQ, units, observation posts, installations and equipment, to the process. This step is not mandatory, but it is our experience that this will ease the RGMP modeling process
2. for every main process elicit the requirements to the processes. Start with the customer processes, then support processes and last rule processes
3. for every requirement find the affected achievers
4. assign one goal to each achiever or group of achievers. Every goal should be measurable and every goal should be aimed at meeting the requirement. Each goal might be divided into sub-goals
5. for every goal use the brain storming technique to find possible problems and means
6. from the goals, problems and means derive requirements for the next process, eg from the goals, problems and means of the customer processes derive requirements for the support processes.

Note that stated requirements could be related to the process itself, its results or its equipment.
7.4.5 Example

In this section an example from the domain of artillery is shown. Note that stated requirements, goals, problems and means are not taken from the Norwegian artillery. They are the author's.

The major contributors of the firing process are shown in Figure 7.18.

![Figure 7.18: Contributors of the firing process.]

Customers of the artillery, eg infantry and armor units, have the following high level requirements to the customer process, ie the firing process, its results and its equipment:

- flexibility
- range: up to 50 km
- at once
- when needed
- right place
- right effect.

Figure 7.20 shows the RGPM model for the requirement *at once*. To be able to meet that requirement the goal "ready to fire one minute after received "call for fire?"" is assigned to the fire control center in the artillery battalion. This goal is divided into three sub-goals: "max 20 seconds to control "call for fire?"", "max 10 seconds to assess "call for fire?"" and "max 30 seconds to make ammunitions and guns ready". These sub-goals are assigned to the fire control center (FCC) in the artillery battalion, the FCC in the artillery batteries and to the gun groups respectively. One goal is also assigned to electronic installations and observation posts, ie artillery locating radar's, forward observers and static forward observers: "ready to transmit "call for fire?" 2 min after the target is detected". Possible problems and means are shown in Figure 7.20.

The initial goal (ready to fire one minute after received "call for fire?") is assigned to the FAbn/FCC because it is the FAbn that receives and responds to the firing request from the forward observers/static forward observes/artillery locating radar's (Figure 7.19). However, the request must also be handled by the FCC in one or more FA batteries and by the involved gun groups before the FAbn/FCC can respond with the message "Ready!" (Figure 7.19).
Figure 7.20: Customer requirement: at once.

7.5 The process sub-language

7.5.1 Introduction

Our process sub-language is inspired by data flow diagram (DFD) languages. The modeling constructs and the principle of using the constructs to build data flow
diagrams are usually easily understood by information system users (ie domain experts). We may create traditional data flow diagrams with our process language, but we have proposed additional user oriented constructs which give us the possibilities to make more expressive process models.

7.5.2 The process language

The process language has five modeling concepts:

- process
- information store
- agent
- message
- co-ordination.

Processes. Normally, processes transform a set of input messages to a set of output messages, but in our domain language some processes have no input messages at all, eg manual order processes which outputs fragmentary orders.

The syntax of a process description has the following characteristics (Figure 7.21):

- *input* on the left hand side
- *output* on the right hand side
- *co-ordination* in the bottom
- *size* indicator showing at which HQ level the process is executed
- *P no*, process number, showing a unique process identifier
- *HQ* showing in which type of headquarters the process is executed
- *process name* gives a short description of what the process is doing
- *center* showing which center in the headquarters that is executing the process
- *role* showing which role in the center (in the headquarters) that is responsible for executing the process.

![Figure 7.21: Process.](image)

It is mandatory to use *P no*, *process name*, *HQ*, and *output*.

For example, in Figure 7.22 it is shown that the operations officer (S-3) in the operations center (OPS) in an artillery battalion HQ is responsible for the execution of
the fire planning process.

![Diagram](image)

**Figure 7.22: Process example.**

We distinguish between general processes, manual processes, IT supported processes and automatic processes as shown in Figure 7.23.

![Diagram](image)

**Figure 7.23: Different types of processes.**

**Information stores** have the same meaning as stores in a DFD. We distinguish between general information stores, paper stores and digital stores as shown in Figure 7.24. To avoid crossing lines in process models, stores may be duplicated (eg Figure 7.35).

<table>
<thead>
<tr>
<th>S no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information store</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Paper store</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Digital store</td>
</tr>
</tbody>
</table>

**Figure 7.24: Different types of stores.**

**Agents** are outside the modeled processes but they are indeed involved in them because they are suppliers and customers of the modeled processes, ie agents might send messages to and/or receive messages from processes. Agents are either headquarters, commanders or non-composite units like observation posts or installations (Figure 7.25). To avoid crossing lines and noisy models, agents are duplicated (eg Figure 7.35, see also subsection 7.5.6 Comments).
Agents with gray fill, e.g. as shown in Figure 7.41, tell readers that these agents are either usually not involved in the modeled process at all or usually not involved at that particular HQ level. The message sequence model of the process in question will show which agents that are not commonly involved in the modeled process and which are not directly involved in the process at that particular level.

**Messages** are input to or output of processes. A message is displayed as a solid arrow where the arrowhead is filled with black as shown in Figure 7.26 a). A message's name and/or distribution mode is optionally described, as shown in Figure 7.26 b) and c) respectively. Messages to or from stores are not named.

![Diagram of message types](diagram)

**Figure 7.26:** Messages.

If an agent or a process receives more than one type of messages from the same source, then the messages are abstracted to just one message. The abstraction is displayed as a bold solid arrow as shown in Figure 7.27 a). At the lowest level of decomposition there are no abstracted messages.

If an agent or a process sends the same message to more than one receiver, just one output is shown but the flow is split just after leaving the process or agent. A dot is used to mark the split. This is shown in Figure 7.27 b).

![Diagram of combined messages](diagram)

**Figure 7.27:** Combined messages.

If two or more messages (arrows) are terminated at the same input point, then only one of the messages are received in a particular situation, i.e. a logical XOR\(^{17}\) as displayed in

---

\(^{17}\) XOR: exclusive OR.
Figure 7.28 a).

If two or more messages are connected to the same output point of a process, then only one of the messages are sent in a particular situation, ie a logical XOR (Figure 7.28 b)).

a) XOR on the input side  b) XOR on the output side

Figure 7.28: Logical XOR.

Logical AND is also supported, but only at the lowest level of decomposition. See subsection 7.5.5 Method for more details.

There are four distribution modes and five message classes.

The four distribution modes are specified by icons that are connected to arrows. The distribution modes are:

- **voice, order meeting:**
- **voice, radio or telephone:**
- **messenger:**
- **electronic:**

The distribution mode *electronic* indicates formatted or free text messages transmitted by computer systems, hand held terminals (eg the Hugin terminal used by the Norwegian mortar units), the Norwegian Multi Role Radio, etc.

The distribution modes *voice, order meeting* and *voice, radio or telephone* signifies synchronous message passing and it is reasonable to think that these two distribution modes indicate that a synchronous interaction is taking place, but that is not quite true. The message passing is strictly one way, but the nature of these two distribution modes make it possible to ask clearing questions. In our domain language a solid, unidirectional arrow is used to indicate who is sending messages to whom and the modeler may add the preferred distribution mode. A bi-directional arrow with broken line is used to indicate interaction, ie co-ordination.

Most messages may be distributed in all four ways. As a rule the most commonly used distribution mode is indicated in the models because the other modes are regarded as back-ups.
The five message classes are:

- *orders*, identified by the syntactical maker "!", eg "fire!"
- *feedback*, also identified by the syntactical maker "!", eg "ready!"
- *requests*, identified by "?", eg "call for fire?"
- *reports*, identified by "report" or "status", eg "ammo status", "shelling report", etc.
- *other messages*. They have no syntactical maker, eg "fire plan".

This classification is not according to Searle's [Searle79] basic categories of illocutionary acts or to Habermas' [Habermas81] classification of speech acts. This is further discussed in subsection 7.5.6 Comments.

**Co-ordination** implies interaction. Interaction is information flow between two parties, but this information flow is normally not formalized, eg like orders, requests, reports and other messages. We have chosen to only indicate that co-ordination is taking place, and provide no special constructs for the actual modeling of the co-ordination process.

Co-ordination is shown as a bi-directional arrow with broken line and with no text as shown in Figure 7.29. Broken lines between units are usually used in the Norwegian Army to indicate co-ordination.

![Co-ordination](image)

Figure 7.29: Co-ordination.

### 7.5.3 Metamodel

The metamodel of the process sub-language is depicted in Figure 7.30.
Figure 7.30: The metamodel of the process sub-language.
7.5.4 Views

The process sub-language might be used in two different ways:

- to describe/prescribe the processes executed in a HQ, eg as shown in Figure 7.31. The focus is on headquarters, ie HQ view. The HQ view should be used to analyze the workload, effectiveness and efficiency of headquarters.
- to describe/prescribe workflows which cross HQ boundaries as shown in Figure 7.32. The focus is on workflow, ie workflow view. The workflow view should be used to analyze the effectiveness and efficiency of workflows.

Note that process models should be designed to meet the goals stated in the RGPM model, eg the workflow view shown in Figure 7.32 should meet the goals stated in Figure 7.20. In Figure 7.20 it is among others stated that the treatment of the request "Call for fire" in the FAbn/FCC and the FAAbt/FCC should not take more than 20 seconds and 10 seconds respectively.

In Figure 7.32 it is assumed that the request for fire is accepted by both the FAbn/FCC and the FAAbt/FCC, ie exception handling is not indicated.

Figure 7.31: HQ view.
Figure 7.32: Workflow view.
7.5.5 Method

Choose one of the main processes from the process overview model and the HQ level (eg battery, battalion, etc) you want to investigate. If you want to investigate a particular workflow, choose the relevant HQ levels. Fill in the fields center and role (eg Figure 7.33). Size, center and role are optional.

![Diagram of fire planning process at the battalion level.]

Figure 7.33: The fire planning process at the battalion level.

Then proceeds as follows:

1. Find agents sending messages to or receiving messages from the main process.

2. Decompose the main process at least until:

   - no sub-process receives more than one message type from agents (eg the message type Fire plan data to process no. 33.11 in Figure 7.34)
   - no sub-process sends more than one message type to agents (eg the message type Fire plan data from process no. 33.12 in Figure 7.34)
   - no sub-process receives messages from both other sub-processes and agents (eg Figure 7.34)
   - no sub-process sends messages to both other sub-processes and agents (eg Figure 7.34).

![Diagram of lowest level of the decomposition, part of the fire planning process.]

Figure 7.34: Lowest level of the decomposition, part of the fire planning process.

It is regarded as normal to go back and forth between the different level of decomposition during the modeling process.
At the lowest level of process decomposition:

- every process should only send one message type to other processes or agents. Some processes may output more than one message type, but just one of the messages should be sent in a particular situation (process no. 13.11 in Figure 7.35). With aid of the port concepts used by PPP (see subsection 6.2.1), this could have been modeled by the XOR port as shown in Figure 7.36.

- every process should only receive one message type from other processes or agents. Some processes may receive more than one message type, but just one of the messages should be received in a particular situation (process no. 13.14 in Figure 7.35). With aid of the port concepts used by PPP (see subsection 6.2.1), this could have been modeled by the XOR port as shown in Figure 7.36.

- indicated message flow between processes and stores should always take place (Figure 7.35). With aid of the port concepts used by PPP (see subsection 6.2.1), this could have been modeled by the AND port as shown in Figure 7.36.

- there should be no abstracted messages (Figure 7.35).

![Diagram of process decomposition](image)

Figure 7.35: Lowest level of decomposition, part of the firing process.

If a process model shows processes that seem to belong to other main processes than the one in focus, e.g., the *Ammunition control* process shown in Figure 7.42 where fire planning is in focus and not logistics, it is reasonable to believe that there is a redesign potential.
7.5.6 Comments

Agents
There are normally several instances of an agent, but in process models it is not indicated if it is the same instance of an agent that sends and receives messages or not. In Figure 7.34 the battery on the left hand side is not the same as the battery on the right hand side, while in Figure 7.35 the battery and the FO on the left hand side are the same as the battery and the FO on the right hand side. We could have chosen to explicitly show if it is the same instance of an agent that sends and receives messages, but that would normally have resulted in noisy models, eg as shown Figure 7.37. From the context users and other people, eg IS engineers, with some domain knowledge easily distinguish if it is the same instance of an agent that sends and receives messages or not.

In process models it is neither explicitly shown if it is just one instance of an agent or several instances of the agent that send or receive messages (eg Figure 7.35) because it is dependent on the situation. For example, a battalion fire mission involves all the batteries in the battalion while a battery fire mission only involves one battery.
Figure 7.37: A model that shows if it is the same instance of an agent that sends and receives messages.

**Sequences**

Process execution sequences might be indicated by means of triggers in process models, but that will normally result in noisy models, eg as indicated in Figure 7.38. Other solutions are:

- to explicitly express the process execution sequences in the text accompanying the process model
- to explicitly show the process execution sequences as a kind of footnote in the process models. The footnote of the process model in Figure 7.31 is shown in Figure 7.39. The time goes from left to right
- to include a process execution sequence view in our domain language. By means of a proper tool, this view could have been implemented as a separate process model view. Thus we could have switched between an information flow view and an
execution sequence view at will. An example of a process execution sequence view is shown in Figure 7.40.

Figure 7.38: Triggers used to indicate the execution sequence.

Figure 7.39: Execution sequence.
Classification of messages
We have classified the messages as follows:

- **orders**, identified by the syntactical marker "!", eg "fire!"
- **feedback**, also identified by the syntactical marker "!", eg "ready!"
- **requests**, identified by "?", eg "call for fire?"
- **reports**, identified by "report" or "status", eg "ammo status", "shelling report", etc.
- **other messages**. They have no syntactical maker, eg "fire plan".

This classification is not according to Searle's [Searle79] basic categories of illocutionary acts:

- **assertives**, commit the speaker to the truth of the expressed proposition, eg "It is raining in Trondheim"
- **directives**, attempts by the speaker to get the hearer to do something, eg "Close the door". Questions are considered to be a subclass of directives
- **commissives**, commit the speaker to some future course of action, eg "I will be there"
- **expressives**, express the psychological state specified in the sincerity condition about a state of affairs specified in the propositional content, eg "Congratulations"
• declaratives, successful performance guarantees the correspondence between the proposition and the world, eg "The ball is out".

Table 7.1 illustrates the differences between our message classes and Searl's speech acts theory. The columns in the table represent Searl's basic categories of illocutionary acts while the rows represents our message classes. The filled rectangles represent similarities.

Reports, eg "ammo status" and "shelling report", and other messages like "Fire plan" and "Coordinates" correspond to Searl's assertives because these messages commit the speaker (sender) to the truth of the message content. Orders, eg artillery operation order and "Fire!", and requests, eg "call for fire?" and "we need food!", correspond to Searl's directives because these messages are attempt by speakers to get the hearer to do something. Feedback, eg "fire plan ready!" and "call for fire denied!", correspond to Searl's declaratives because these messages guarantees the correspondence between the proposition, ie message content, and the world, ie the actual situation.

<table>
<thead>
<tr>
<th>Our Message</th>
<th>Assertives</th>
<th>Directives</th>
<th>Commissives</th>
<th>Expressives</th>
<th>Declaratives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reports</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of our message classes and Searl's speech acts theory.

Our message classification is neither according to Habermas' [Habermas81] classification of speech acts (we use the same terms as in [Dietz91]):

• imperativa, the sender aims at a change of state in the objective world and attempts to let the hearer act in such a way that this change is brought about, eg "Shut up". The dominant claim is the power claim
• constativa, the sender asserts something about the state of affairs in the objective world, eg "It is raining in Trondheim". The dominant claim is the claim to truth
• regulativa, the sender refers to a common social world in such a way that he tries to establish an interpersonal relation which is consider to be legitimate, eg "Close the door, please". The dominant claim is the claim to justice
• expressiva, the sender refers to his subjective world in such a way that he discloses publicly a lived experience, eg "Congratulations". The dominant claim is the claim to sincerity.

Table 7.2 illustrates the differences between our message classes and Habermas' speech acts theory. The columns in the table represent Habermas' classification of speech acts while the rows represents our message classes. The filled rectangles represent similarities.
Orders, eg artillery operation order and "Fire!", correspond to Habermas’ imperativa because the objective of these messages are to change the state in the objective world and to let the hearer act in such a way that this change is brought about. The dominant claim is indeed the power claim.

According to the comparison of Searle’s and Habermas’ taxonomy of speech acts in [Dietz91], commands are regulativa while expressions of will are imperative. Military commands, and thus orders, are expressions of will. Thus, the dominant claim is the power claim and not the claim to justice. We therefore claim that orders correspond to Habermas’ imperativa.

In [Dietz91] it is claimed that Habermas’ regulativa encompass entirely Searle’s declaratives. Thus our feedback messages, eg "fire plan ready!" and "call for fire denied!", should correspond to Habermas’ regulativa.

Requests, eg "call for fire?" and "we need food!", correspond to Habermas’ regulativa because these messages tries to establish an interpersonal relation which is consider to be legitimate. The dominant claim is the claim to justice. Reports, eg "ammo status" and "shelling report", and other messages like "Fire plan" and "Coordinates" correspond to Habermas’ constativa because these messages assert something about the state of affairs in the objective world.

<table>
<thead>
<tr>
<th>Habermas</th>
<th>Imperativa</th>
<th>Constativa</th>
<th>Regulativa</th>
<th>Expressiva</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reports</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of our message classes and Searl’s speech acts theory.

Our classification of messages is not based on any linguistic theory. It is a pragmatic classification. There is a huge difference between orders and requests in a military system, thus they should be treated as two different message classes. There is also a huge difference between feedback and requests in a military system. The two message classes reports and other messages are indeed assertives/constativa, but we find it useful to distinguish between reports which are descriptive (ie looking backward) and messages that are more prescriptive (ie looking forward), eg the message "fire plan".

7.5.7 Example, Fire planning

The process "Fire planning" at the battalion level is carried out in the operations center (OPS) in the artillery battalion HQ (Figure 7.41). The operations officer (S-3) is responsible for the execution of the process. The fire planning process is IT supported. The artillery operation order (Art. ops!) is received at an order meeting while the fire
plan data from the battery commander (BC) is received electronically. The message "Fire plan ready!" from the reinforce artillery battalion HQ (Rbn) is received by radio or telephone. The process receives some messages from the artillery battery HQ, but at this process level it is not explicit stated which messages that are received. The fire planning process outputs a lot of messages but just one is explicit stated, the "Fire plan ready!" which is transmitted to the BC by radio or telephone.

Figure 7.41: The fire planning process, first battalion level

At the second battalion level, the fire planning process shown in Figure 7.41 is decomposed into five processes, as shown in Figure 7.42. The "Ammunition control" process is co-ordinated with the "Ammunition management" process in the artillery battalion's administration center (ADM). The "Orders" process outputs fragmentary orders, eg which meteorological data should be used to calculate the gunnery data. In Figure 7.41 it is seen that the reinforce artillery battalion (Rbn) receives an abstracted message from the "Fire planning" process. In Figure 7.42 this abstracted message has been decomposed into "Fire plan data" and "Coordinates". Note that the operation officer (S-3) is responsible for the execution of the overall fire planning process shown in Figure 7.41 while the duty officer (DO) is responsible for the execution of three of the sub-processes shown in Figure 7.42. S-3 has delegated some of the responsibilities to the DO.
At the third battalion level, the fire plan management process shown in Figure 7.42 is decomposed into six processes, as shown in Figure 7.43. The process "Receive and store fire plan data" is automatic and thus executed by a computer (WS: work station). This is the last level of decomposition. Every process has only one type of external input and external output, and there are no abstracted messages.
7.6 The message sequence sub-language

7.6.1 Introduction

Information and distribution of relevant information are of utmost importance in executive oriented information system like military CCIS'. To visualize the information exchange between headquarters, commanders, observation posts and installations, we
will use message sequence models. Message sequence models could also be used to visualize information exchange between centers in a HQ, or between roles in a HQ.

Note that the message passing sequence is not strict. A message sequence models just indicates what is regarded as a normal sequence, for example the order "Target recording!" in Figure 7.47 may arrive before the order "Distribution of ammo!". Thus a message sequence model just shows a sample message sequence.

Our message sequence sub-language is inspired by the message sequence language recommended by CCITT [CCITT120] and by the scenario language in OOram [Reenskaug95].

7.6.2 The message sequence language

The message sequence language has two modeling concepts:

- agent
- message.

**Agents** send messages to each other. Agents are shown at the top of the model, and they are either headquarters, commanders or non-composite units like observation posts or installations (Figure 7.44). Note that agents in this sub-language have the same meaning as agents in the process sub-language.

![Artillery battery Forward observer Battery commander](image)

**Figure 7.44: Example of agents.**

Agents with gray fill, eg as shown in Figure 7.47, tell readers that these agents are usually not involved in the modeled process at all.

A vertical timeline is connected to each agent, and it shows where arrows should start and end.

**Messages**, message classes, conditions (rules), and comments are shown at the left hand side of the model. Arrows represent messages, and point from senders to receivers of messages.

Several arrows may start at the same point. This means that the message is distributed to every receiver, ie a logical AND (Figure 7.45). A dot is used to indicate the starting point of the message.
The message sequence language has the same five message classes and four distribution modes as the process sub-language has. It is optional to model distribution modes.

To repeat, the message classes are:

- **orders**, identified by the syntactical maker "!", for example "fire!"
- **feedback**, also identified by the syntactical maker "!", for example "ready!"
- **requests**, identified by "?", for example "call for fire?"
- **reports**, identified by "report" or "status", for example "ammo status", "shelling report", etc.
- **other messages**. They have no syntactical maker, for example "fire plan".

The four distribution modes are specified by icons that are connected to arrows. The distribution modes are:

- **voice, order meeting**:  
- **voice, radio or telephone**:  
- **messenger**:  
- **electronic**:  

Some messages are conditional, i.e. one or more rules are connected to those messages. Simple rules are explicitly stated in the message sequence models. The rules are shown in brackets, e.g. "Fire plan [Brigade: if no. infbn > 1]"\(^{18}\). Complex rules are not explicitly stated, but all conditional messages and associated rules are stated in the message rules table.

Note that icons and arrows together with the AND construct express rules too. Icons explicitly state how messages normally should be distributed while arrows explicitly state which messages agents should send to other agents.

Comments are shown in parentheses, ( ).

The syntax of the message sequence language is shown in Figure 7.47.

In message sequence models it is not explicitly shown if it is just one instance of an agent or several instances of the agent that send or receive messages because it is

---

\(^{18}\) The condition says that the message is only sent to the brigade if the number of infantry battalions (infbn) involved in the fire plan is greater than one.
situation and context dependent. From the context users and other people, eg IS engineers, with some domain knowledge are able to tell if one or more instances are involved in the process. For example, in the fire planning process shown in Figure 7.47 all the instances of the shown agents are involved.

7.6.3 Metamodel

The metamodel of the message sequence sub-language is depicted in Figure 7.46.

![Diagram of message sequence sub-language metamodel]

Figure 7.46: The metamodel of the message sequence sub-language.

7.6.4 Example

Figure 7.47 shows a message sequence model of the fire planning process. Messages are shown at the left hand side of the model while agents are shown at the top. The message sequence model in Figure 7.47 shows which message is sent to which agent during the fire planning process.
Figure 7.47: Message sequence model of the fire planning process.

7.7 The role sub-language

A role model, created by aid of the role language, shows the headquarters organization, ie centers and roles included in each center.

7.7.1 The role language

The role language has three modeling concepts:

- headquarters
- center
- role.

**Headquarters.** A HQ is represented by its military symbol, eg an armored artillery
battalion HQ (Figure 7.48). A HQ is composed of centers.

![Figure 7.48: An armored artillery battalion HQ.](image)

**Center.** A center is represented by its abbreviation, and for modeling simplicity the abbreviation is put into a rectangle, eg OPS (operations center) as shown in Figure 7.49. A center is composed of roles and a center is part of a HQ.

![Figure 7.49: OPS.](image)

**Role.** A role is represented by its abbreviation, and for modeling simplicity the abbreviation is put into a rectangle, eg S-3 (operations officer) as shown in Figure 7.50. A role is part of a center.

![Figure 7.50: S-3.](image)

Aggregation between the concepts is displayed as a branched line. The branch is horizontally between the HQ and the centers and vertically between a center and the roles (Figure 7.51 a)). A HQ consists of one or more centers and a center consists of several actors playing different roles. Cardinalities are used to indicate the number of equal centers and roles. A number is used to specify cardinalities greater than one (Figure 7.51 b)).

The syntax of the role language is shown in Figure 7.53.

![Figure 7.51: Aggregation and cardinalities.](image)

a) Aggregation  
b) Cardinalities
7.7.2 Metamodel

The metamodel of the role sub-language is depicted in Figure 7.52.

![Diagram of metamodel]

Figure 7.52: The metamodel of the role sub-language.

7.7.3 Example

Figure 7.53 shows that an armored artillery battalion HQ has two centers: OPS (operations) and ADM (administration), and that the operations center (OPS) has the

![Diagram of role model]

Figure 7.53: Role model.
following roles: one S-2 (intelligence officer), one S-3 (operations officer), three Ass S-3/DO (S-3 assistants which also are duty officers (DO's)), three HQ ass (Headquarters assistants), six RO's (radio operators) and one WS (work station, ie computer).

7.8 **The organization structure sub-language**

7.8.1 **Introduction**

The organization structure sub-language we are using in our domain language is based on the one commonly used in the Norwegian army. We use this language to build organization structure models. An organization structure model is a traditional hierarchy showing a unit and its sub-units.

7.8.2 **The organization structure language**

The organization structure language has two modeling concepts:

- unit
- headquarters

**Unit.** In this context a unit includes composite units and non-composite units like observation posts and installations (Figure 7.56). Units are represented by military symbols as defined in chapter 3.

**Headquarters** are in this context not represented by military symbols as defined in chapter 3 but by the abbreviation HQ. In the context of organization structure models this abbreviation is unambiguous because organization structure models explicitly shows which unit a HQ is part of, as illustrated in Figure 7.54.

![Diagram](image)

**Figure 7.54:** HQ is part of unit.

Composite units might be composed of other composite units and/or non-composite units and headquarters as shown in the metamodel (Figure 7.56).

To facilitate the readability of organization structure models, we strongly recommend to only view or present three levels of an organization structure model simultaneously.
The aggregation is displayed as a branched line. We recommend a horizontal branch between the top level and the second level and a vertical branch between the second level and third level (Figure 7.55 a)). Cardinalities are used to indicate the number of equal units. A number is used to specify cardinalities greater than one (Figure 7.55 b)).

Figure 7.55: Aggregation and cardinalities.

7.8.3 Metamodel

The metamodel of the organization structure sub-language is shown in Figure 7.56.

Figure 7.56: Metamodel of the organization structure sub-language.
### 7.8.4 Example

Figure 7.57 shows an example of an organization structure model for an armored artillery battalion. An artillery battalion consists of three or four artillery batteries, one staff battery and the battalion HQ. An artillery battery consists of the battery HQ and of the following platoons: staff, gun, forward observer, signal and train. The staff battery consists of the battery HQ, a meteorological group and the following platoons: staff, static forward observer, artillery locating radar, survey, signal, medical, ammunition, maintenance and train.

![Diagram of organization structure model]

Figure 7.57: Three levels of an organization structure model.
7.9 Integrated metamodell

In Figure 7.58 and Figure 7.59 a complete metamodell showing the integration of the different sub-languages in to a compound language is presented.

Figure 7.58: Complete metamodell, part 1.
Figure 7.59: Complete metamodel, part 2.
7.10 Inter-model relations

The most used relations between models are shown in Figure 7.60.

![Diagram showing inter-model relations]

**Figure 7.60: Inter-model relations.**

**Relations between organization structure models and role models**

The relations between organization structure models and role models are shown in Figure 7.61. Note that a unit and its HQ make use of the same role indicator (see chapter 3), but to avoid confusions in the organization structure models headquarters are denoted HQ.

![Diagram showing relations between organization structure models and role models]

**Figure 7.61: Relations between organization structure models and role models.**
Relations between process models and process overview models
The relations between process models and process overview models are shown in Figure 7.62. The Process number (P. no) of a process should be correlated with the corresponding P. no of the overview process and the units should be equal. In addition, the top level process of a process model should have the same process name as the corresponding overview process.

![Diagram showing relations between process models and process overview models.]

Figure 7.62: Relations between process models and process overview models.

Relations between process models and message sequence models
As depicted in Figure 7.63 a message sequence model might be related to many process models while a process model should not be related to more than one message sequence model.

Most of the main processes, e.g., the fire planning process, take place at several HQ levels. A process model is made for each HQ level\(^\text{19}\) while just one message sequence model covering all levels is made. Thus a main process is described by one message sequence model and several process models.

![Diagram showing relation between a message sequence model and process models.]

Figure 7.63: Relation between a message sequence model and process models.

It is the concepts of message and agent that connect the message sequence model and the process models as shown in Figure 7.64 (to avoid a noisy figure only one of the relations are explicitly shown by an arrow). Every external message (i.e., messages sent to or received from agents) and every agent shown in a process model should also be

\(^{19}\)It might be a better idea to just talk about one process model for each of the main processes, from where HQ views and workflow views are created, but such a solution requires sophisticated tool support including a suitable repository solution.
shown in the corresponding message sequence model. In addition the HQ level modeled by a process model should be shown as an agent in the corresponding message sequence model.

![Diagram](image_url)

Figure 7.64: Relations between process models and message sequence models.

**Relations between process models**
As mentioned above one process model is made for each HQ level. The relations but one between process models are shown in Figure 7.65. The relation that is not indicated in Figure 7.65 is the relation between process numbers. The process numbers of related process models should be correlated, e.g., in Figure 7.65 the first digit of the process number represents the modeled main process.
Figure 7.65: Relations between process models.

The relations shown in Figure 7.65 and the process number relation could also be used to create workflow views on the basis of HQ views. The workflow view in Figure 7.66 is created on the basis of the two HQ views shown in Figure 7.65.

Figure 7.66: Workflow view.

**Relations between process models and role models**
The relations between process models and role models are shown in Figure 7.67.
Relations between process models and organization structure models
The relations between process models and organization structure models are shown in Figure 7.68\textsuperscript{20}.

Relations between message sequence models and role models
Relations between message sequence models and role models are shown in Figure 7.69.

\textsuperscript{20} Note that a unit and its HQ make use of the same role indicator (see chapter 3), but to avoid confusions in the organization structure models headquarters are denoted HQ.
Figure 7.69: Relations between message sequence models and role models.

Relations between message sequence models and organization structure models
The relations between message sequence models and organization structure models are shown in Figure 7.70.

Figure 7.70: Relations between message sequence models and organization structure models.
Relations between RGPM models and process overview models
The relations between process overview models and RGPM models are shown in Figure 7.71.

Figure 7.71: Relations between RGPM models and process overview models.

Relations between RGPM models and role models and organization structure models
The relations between RGPM models and role models and organization structure models are shown in Figure 7.72.
Figure 7.72: Relations between RGPM models and role models and organization structure models.

7.11 Tables

Tables are tabular views of some of the information collected and stored in the repository. There are four information tables in our domain language:

- the message content table
- the responsibility table
- the rule table
- the goal table.

Figure 7.73 shows the relations between models and tables. Tables are displayed as rectangles while models are displayed as rectangles with rounded corners. An arrow has the meaning "is generated on the basis of", i.e. it shows from which models a table might be generated on the basis of. For example, the responsibility table might be generated on the basis of a process model or a role model, and the goal table might be generated on the basis of the RGPM model.
7.11.1 The message content table

The objective of the message content table is to give a detailed overview of what information is needed by a process and what information it has to produce (ie output) in order to meet input demands from other processes or agents. The message content table is a tabular view of information needed by a process.

The message content table is a matrix as shown in Table 7.3 and Table 7.4. In the top row the column "PROCESS" in focus is shown between the "Input from" and the "Output to" columns. Agents, stores and/or processes sending messages to the process in focus are shown at process’ left hand side while agents, stores and/or processes receiving messages from the process in focus are shown at its right hand side. There should be one column for each input and each output. The message content is shown right bellow the top row. The message’s name is shown if it is a message to or from an agent.

For some messages the contents are abstracted, eg the Target recording message shown in Table 7.3. This means that detailed information could be found in other sources, eg forms, order templates, etc. Abstracted message content is indicated with aid of a bullet, *, in front of each item.

For other messages the content is detailed, eg the message outputted to the store Coordinate table in Table 7.3. Detailed message content is indicated with aid of the symbol "<" in front of each item.
Table 7.3: Information needed by the process Calculate coordinates.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
</table>
| Target recording  
  * type of order  
  * type of target location  
  * target location | 34.12 |
| S34.2 Coordinate table |
| < target number  
  < east  
  < north  
  < altitude |

Table 7.4: Information needed by the process Circulating fire plan data.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
</table>
| Fire plan data  
  < target number  
  < from time  
  < to time  
  < east  
  < north  
  < registered by  
  < registered with  
  < ammo type  
  < type of target  
  < no. of batteries  
  < duration  
  < no. of targets  
  < comments | 33.12 |
| R |
| Fire plan data:  
  < target number  
  < from time  
  < to time  
  < east  
  < north  
  < registered by  
  < registered with  
  < ammo type  
  < type of target  
  < no. of batteries  
  < duration  
  < no. of targets  
  < comments |

By means of a proper tool it should be possibility to select a message in a process model or a message sequence model and then specify that one want to enter, edit or view the message's content, as illustrated in Figure 7.74.
Message content tables are generated on the basis of process models.

### 7.11.2 The responsibility table

The responsibilities (i.e., processes) of headquarters, centers, and roles could be listed in a responsibility table as shown in Table 7.5. "P. no", i.e., process number, is a reference to the process in the process model. "P. no" is especially useful in case of no tool support.

Responsibility tables are generated on the basis of process models and/or role models. The responsibility table shown in Table 7.5 is generated on the basis of a process model and the corresponding role model. The columns HQ, centers, and roles in Table 7.5 are filled on the basis of the role model while the remaining columns are filled on the basis of the process model. Responsibilities could also be filled in manually.

Only top-level processes are listed for headquarters and centers.

<table>
<thead>
<tr>
<th>HQ</th>
<th>Centers</th>
<th>Roles</th>
<th>Responsible for process</th>
<th>P. no</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>OPS</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>S-2</td>
<td>Fire planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ammunition control</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receive and store the artillery operation order</td>
<td></td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td>Ass S-3/DO</td>
<td>Fire plan management</td>
<td></td>
<td>33.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circulating fire plan data</td>
<td></td>
<td>33.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Record fire plan data in concentration table</td>
<td></td>
<td>33.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete fire plan data from data store</td>
<td></td>
<td>33.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distribute fire plan</td>
<td></td>
<td>33.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Report fire plan ready</td>
<td></td>
<td>33.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinates management</td>
<td></td>
<td>33.2</td>
</tr>
<tr>
<td></td>
<td>HQ ass</td>
<td>Orders</td>
<td></td>
<td>33.5</td>
</tr>
<tr>
<td></td>
<td>RO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processes</td>
<td>Rules</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33.12</td>
<td>When fire plan data is received and stored</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Circulate fire plan data</td>
<td>Then circulate the fire plan data to the R-FAbn and to all FAbtt's but the one that sent the message</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPS, DO</td>
<td>and trigger off process no 33.13 Record fire plan data in the fire plan table</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.6: Rule table, processes.

<table>
<thead>
<tr>
<th>Messages</th>
<th>From</th>
<th>To</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make ammo ready!</td>
<td>FAbn</td>
<td>FAbtt</td>
<td>Attack only</td>
</tr>
</tbody>
</table>

Table 7.7: Rule table, messages.

By means of a proper tool it should be possibility to select a process in a process model or a message in a message sequence model and then specify that one want to enter, edit or view the rules, eg as illustrated in Figure 7.75.
7.11.4 The goal table

Goals assigned to headquarters, units and/or equipment could be listed in a goal table as shown in Table 7.8.

Goal tables are generated on the basis of RGPM models.

<table>
<thead>
<tr>
<th>Achievers</th>
<th>Assigned goals</th>
</tr>
</thead>
</table>
| FCC       | ready to fire 1 min. after received "call for fire?"  
           | max 20 sec to control "call for fire?" |
|           | ready to transmit "call for fire?" 2 min after the target is detected |
|           | ready to transmit "call for fire?" 2 min after the target is detected |
|           | ready to transmit "call for fire?" 2 min after the target is detected |
|           | max 30 sec to make ammo and guns ready |

Table 7.8: Goal table.

7.12 Method

In this section we describe how to use the different sub-languages in relation to each other in various situations.

Our domain language could be used to model the current state (ie organization, goals, processes, roles, information and IT) of the domain, a desired state, a temporary state, or what ever. That is, our domain language is not restricted to be used in a particular situation.
Normally, a modeling language is used to:

- model, ie document, the current state
- model a desired state
- model both the current state and a desired future state.

We recommend to first model the current state of the domain and then if deemed necessary the desired future state. Normally, it is a good idea to find out where you are before you decide where to go.

### 7.12.1 Modeling of the current state

We recommend to always have updated models of the current state of the domain because these models give a good description of the domain’s structure, working processes, etc. These models could be used to educate new employees and they constitute a valuable starting point when changes are to be made to the domain.

Organization structure models and role models give a good overview of the units, headquarters, centers and roles included in the domain. Therefore, we recommend the analysts (either they are users, professional analysts like IS engineers or preferably both) to start to analyze and model the organization structure and the headquarters organizations, ie centers and roles. When the organization structure and roles are modeled, we recommend the analysts to get an overview of the main processes executed in the domain, ie to make a process overview model.

![Diagram](image)

Figure 7.76: Modeling approach, current state.
We will not give an explicit recommendation of how to proceed, but we strongly recommend the analysts to decompose the main processes (ie make process models), make message sequence models, build RGPM models and to complete the tables. However, it is our experiences that it is advantageous to do much of these tasks in parallel as illustrated in Figure 7.76.

Even if the above description of the modeling process is sequential, the analysts should feel free to go back and forth between the various stages at will.

7.12.2 Modeling of the desired future state

Before the desired future state of the domain is modeled one ought to have a correct model of the current state of the domain.

It is tempting to advice analysts to forget the organization structure, roles and other structured elements and to design new working processes from scratch, and to derive the new organization on the behalf of the new working processes, ie a true BPR [Hammer 90] approach. Maybe the artillery would have discovered that the battery level is superfluous. However, we will not recommend such an approach because the design of operational military organizations is affected by much more than just working processes, eg a manageable span of maneuvering control and vulnerability considerations.

We recommend an iterative approach:

WHILE not satisfied DO

1. accomplish major structural changes, eg introduction of an additional echelon (eg artillery regiment), or additional units (eg rocket artillery)
2. investigate the main processes. Have every process a mission? Could some processes be eliminated or merged with other processes? Are additional processes required?
3. build a RGPM model for each of the main processes and generate the goal table
4. propagate the changes in the process overview model to the other models and to the tables
5. investigate the message sequences. Many messages indicate that the process is inefficient or ineffective, ie:
   - reduce the number of control messages, eg Prepare ammo!, to a minimum
   - reduce the number of transit messages to a minimum
   - eliminate unnecessary messages
   - investigate if messages could be redirected.

In parallel, corresponding process models and influenced tables should be updated

6. investigate the process models and redesign if deemed appropriate. Additional process should also be designed if required. In parallel corresponding message sequence models, role models, organization structure models and influenced tables
should be updated
7. for every process analyze in details what information it needs and what
information it has to produce (ie output) in order to meet input demands from
other processes or agents. This should be documented in the message content table
8. start at point 1 again or terminate.

END WHILE

Even if the above description of the modeling process is sequential, the analysts should
feel free to go back and forth between the various stages at will.
8 Case Study

In this chapter an almost complete modeling example of the domain of artillery is presented. We have modeled the current state and the desired future state, we have used all the sublanguages and tables, we have modeled the Fire planning process at the battalion level to the utmost, and we have used the method recommended in section 7.12.

Note that shown models and filled tables don't describe the Norwegian artillery as it is or is planned to be. To avoid classified information we have done some modifications to its structures, working process, and so on.

8.1 Current state

8.1.1 Organization structure models

We have chosen to show three organization structures in this case study. Figure 8.1 shows the current organization structure of a field artillery battalion (rotated 90 degrees), Figure 8.2 shows the current organization structure of a field artillery battery while Figure 8.3 shows the current organization structure of a gun group.
Figure 8.1: Field artillery battalion.

Figure 8.2: Field artillery battery.

Figure 8.3: Gun group.
8.1.2 Role models

Figure 8.4 shows the centers and roles of the battalion HQ and battery HQ.

![Diagram showing the roles of battalion HQ and battery HQ]

Figure 8.4: Battalion HQ and battery HQ.

8.1.3 Process overview model

Figure 8.5 shows the current process overview model of the artillery. Since the number of customer processes are higher than one it is reasonable to believe that there is a redesign potential.
8.1.4 Process models

The figures in this subsection (Figure 8.6 to Figure 8.11) describe the current fire planning process at the battalion level.

Figure 8.6: The fire planning process (first level).
The fire planning process is decomposed at least until none of its sub-processes

- receive more than one message type from agents
- send more than one message type to agents
- receive messages from both other sub-processes and agents
- send messages to both other sub-processes and agents.

Figure 8.7: The decomposition of the fire planning process (second level).
Note that process number 33.4 shown in Figure 8.7 is not decomposed any further.

Figure 8.8: The decomposition of the fire plan management process (third and final level).
Figure 8.9: The decomposition of the coordinates management process (third and final level).

Figure 8.10: The decomposition of the ammunition control process (third and final level).

Since the ammunition control processes seems to belong to the logistics process it is
reasonable to believe that there is a redesign potential.

Figure 8.11: The decomposition of the orders process (third and final level).

8.1.5 Message sequence models

Figure 8.12 shows the message sequence model of the current fire planning process.

Figure 8.12: Message sequence model of the fire planning process.
8.1.6 Rule tables

Rules for processes at the lowest level of decomposition are shown in Table 8.1.

<table>
<thead>
<tr>
<th>Processes</th>
<th>Rules</th>
</tr>
</thead>
</table>
| 33.4      | When the artillery operation order (Art.ops) is received  

Then trigger off the fire planning process  

and store the artillery operation order in the artillery operation order archive |

| 33.11     | When the message fire plan data is received  

Then store the fire plan data in the data store  

and trigger off process no. 33.12 Control fire plan data |

| 33.12     | When the fire plan data is received and stored  

Then control the fire plan data against the safety data  

If the fire plan data is judged OK  

Then trigger off process no. 33.14 Circulate fire plan data  

Else trigger off process no. 33.13 Send the message repeat |

| 33.13     | When the fire plan data is not judged OK  

Then send the message repeat to the FAbt that sent the fire plan data  

and trigger off process no. 33.16 Delete fire plan data from data store |

| 33.14     | When the fire plan data is judged OK  

Then circulate the fire plan data to the R-FAbn and to all FAbtt's but the one that sent the message  

and trigger off process no 33.15 Record fire plan data in the fire plan table |

| 33.15     | When the fire plan data is circulated  

Then record the fire plan data in the fire plan table  

and trigger off process no. 33.16 Delete fire plan data from data store |

| 33.16     | When the fire plan data is recorded in the fire plan table  

or the message repeat is sent  

Then delete the fire plan data from the data store |

| 33.17     | When time  

Then  

If the number of maneuvering units involved is greater than one  

Then distribute the fire plan table to both the brigade and the division  

Else distribute the fire plan table to the division |
<table>
<thead>
<tr>
<th>33.18</th>
<th>Report fire plan ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the message fire plan ready is received  
Then  
If two or more batteries and the reinforced FAbn have reported fire plan ready  
Then send the message fire plan ready to battery commanders, the brigade and the division  
Else wait

<table>
<thead>
<tr>
<th>33.21</th>
<th>Store coordinates and notify DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>WS</td>
</tr>
</tbody>
</table>

When the message coordinates is received  
Then store the coordinates in the data store and trigger off process no. 33.22 Control coordinates

<table>
<thead>
<tr>
<th>33.22</th>
<th>Control coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the coordinates are received and stored  
Then control the coordinates against the safety data  
If the coordinates are judged OK  
Then trigger off process no. 33.24 Circulate coordinates  
Else trigger off process no. 33.23 Send the message repeat

<table>
<thead>
<tr>
<th>33.23</th>
<th>Send the message repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the coordinates are not judged OK  
Then send the message repeat to the FAbt that sent the coordinates and trigger off process no. 33.26 Delete coordinates from data store

<table>
<thead>
<tr>
<th>33.24</th>
<th>Circulate coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the coordinates are judged OK  
Then circulate the coordinates to the R-FAbn and to all FAbt’s but the one that sent the message and trigger off process no 33.25 Record coordinates in coordinate table

<table>
<thead>
<tr>
<th>33.25</th>
<th>Record coordinates in coordinate table</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the coordinates are circulated  
Then record the coordinates in the coordinate table and trigger off process no. 33.26 Delete coordinates from data store

<table>
<thead>
<tr>
<th>33.26</th>
<th>Delete coordinates from data store</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When the coordinates are recorded in the coordinate table or the message repeat is sent  
Then delete the coordinates from the data store

<table>
<thead>
<tr>
<th>33.27</th>
<th>Distribute coordinate table</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>DO</td>
</tr>
</tbody>
</table>

When every target listed in the fire plan have recorded coordinates or time  
Then distribute the coordinate table to the division

<table>
<thead>
<tr>
<th>33.31</th>
<th>Make directive regarding distribution of ammo</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPS</td>
<td>S.3</td>
</tr>
</tbody>
</table>

When time  
Then make a directive regarding the distribution of ammunition and distribute it to the batteries
When time
Then give orders to the batteries regarding the preparation of ammunition

When time
Then give orders to the batteries regarding which meteorological data to use when gunnery data is to be calculated

When time
Then give orders to the batteries regarding when to calculate gunnery data

Table 8.1: Rule table, processes.

Rules for conditional messages are shown in Table 8.2.

<table>
<thead>
<tr>
<th>Messages</th>
<th>From</th>
<th>To</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fire plan data (circulate)</td>
<td>FAbn</td>
<td>FAbt R- FAbn</td>
<td>To all FAbt’s but the one that sent the message to the FAbn</td>
</tr>
<tr>
<td>Fire plan</td>
<td>FAbn</td>
<td>Brig Div</td>
<td>Only to the brigade if the number of infantry battalions involved is greater than one</td>
</tr>
<tr>
<td>Prepare ammo!</td>
<td>FAbn</td>
<td>FAbt</td>
<td>In attack only</td>
</tr>
<tr>
<td>Ready!</td>
<td>Gun groups</td>
<td>FAbt</td>
<td>When the prescribed amount of ammunition is prepared and guns in position</td>
</tr>
<tr>
<td>Fire plan ready!</td>
<td>R- FAbn</td>
<td>FAbn</td>
<td>When two or more batteries have reported fire plan ready</td>
</tr>
<tr>
<td>Fire plan ready!</td>
<td>FAbt</td>
<td>FAbn</td>
<td>When the prescribed amount of ammunition is prepared, and every target listed in the fire plan is recorded, and the technical baseline is settled, and two or more gun groups have reported ready</td>
</tr>
<tr>
<td>Fire plan ready!</td>
<td>FAbn</td>
<td>BC Brig Div</td>
<td>When two or more batteries and the reinforced FAbn have reported fire plan ready</td>
</tr>
</tbody>
</table>

Table 8.2: Rule table, messages.
### 8.1.7 Responsibility table

Table 8.3 summarize the responsibilities assigned to HQ, centers and roles.

<table>
<thead>
<tr>
<th>HQ</th>
<th>Centers</th>
<th>Roles</th>
<th>Responsible for process</th>
<th>P. no</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>OPS</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>S-2</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>S-3</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ammunition control</td>
<td></td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Make directive regarding distribution of ammo</td>
<td></td>
<td>33.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Give orders regarding preparation of ammo</td>
<td></td>
<td>33.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receive and store the artillery operation order</td>
<td></td>
<td>33.4</td>
</tr>
<tr>
<td>Ass S-3/DO</td>
<td></td>
<td>Fire plan management</td>
<td></td>
<td>33.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control fire plan data</td>
<td></td>
<td>33.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send the message repeat</td>
<td></td>
<td>33.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circulate fire plan data</td>
<td></td>
<td>33.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Record fire plan data in fire plan table</td>
<td></td>
<td>33.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete fire plan data from data store</td>
<td></td>
<td>33.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distribute fire plan</td>
<td></td>
<td>33.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Report fire plan ready</td>
<td></td>
<td>33.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinates management</td>
<td></td>
<td>33.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control coordinates</td>
<td></td>
<td>33.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send the message repeat</td>
<td></td>
<td>33.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circulate coordinates</td>
<td></td>
<td>33.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Record coordinates in coordinate table</td>
<td></td>
<td>33.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete coordinates from data store</td>
<td></td>
<td>33.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distribute coordinate table</td>
<td></td>
<td>33.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Orders</td>
<td></td>
<td>33.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Give orders regarding which MET data to use</td>
<td></td>
<td>33.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Give orders regarding gunnery data calculation</td>
<td></td>
<td>33.52</td>
</tr>
<tr>
<td>HQ ass</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS</td>
<td></td>
<td>Receive and store fire plan data</td>
<td></td>
<td>33.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receive and store coordinates</td>
<td></td>
<td>33.21</td>
</tr>
<tr>
<td>ADM</td>
<td></td>
<td>Ammunition management</td>
<td></td>
<td>43.1</td>
</tr>
</tbody>
</table>

Table 8.3: Responsibility table.
### 8.1.8 Message content tables

Message content tables show the input and output of processes. Table 8.4 to Table 8.11 show the message content tables for the sub-processes of the fire plan management process (based on Figure 8.8).

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC</td>
<td><img src="image" alt="33.11" /></td>
<td><img src="image" alt="S33.3" /></td>
</tr>
<tr>
<td>Fire plan data</td>
<td></td>
<td>Fire plan data</td>
</tr>
<tr>
<td>&lt; target number</td>
<td></td>
<td>&lt; target number</td>
</tr>
<tr>
<td>&lt; from time</td>
<td></td>
<td>&lt; from time</td>
</tr>
<tr>
<td>&lt; to time</td>
<td></td>
<td>&lt; to time</td>
</tr>
<tr>
<td>&lt; east</td>
<td></td>
<td>&lt; east</td>
</tr>
<tr>
<td>&lt; north</td>
<td></td>
<td>&lt; north</td>
</tr>
<tr>
<td>&lt; registered by</td>
<td></td>
<td>&lt; registered by</td>
</tr>
<tr>
<td>&lt; registered with</td>
<td></td>
<td>&lt; registered with</td>
</tr>
<tr>
<td>&lt; ammo type</td>
<td></td>
<td>&lt; ammo type</td>
</tr>
<tr>
<td>&lt; type of target</td>
<td></td>
<td>&lt; type of target</td>
</tr>
<tr>
<td>&lt; no. of batteries</td>
<td></td>
<td>&lt; no. of batteries</td>
</tr>
<tr>
<td>&lt; duration</td>
<td></td>
<td>&lt; duration</td>
</tr>
</tbody>
</table>

Table 8.4: Process no. 33.11.

<table>
<thead>
<tr>
<th>Input from</th>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="S33.3" /></td>
<td><img src="image" alt="S33.7" /></td>
<td><img src="image" alt="33.12" /></td>
<td><img src="image" alt="33.13" /></td>
<td><img src="image" alt="33.14" /></td>
</tr>
<tr>
<td>Fire plan data</td>
<td>Safety data</td>
<td>Control fire plan data</td>
<td>Send the message repeat</td>
<td>Circulate fire plan data</td>
</tr>
<tr>
<td>&lt; east</td>
<td>&lt; north</td>
<td>• FLOT</td>
<td>• coordination lines</td>
<td>&lt; repeat</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• position of friendly units in front of FLOT</td>
<td>• boundary lines</td>
<td>&lt; OK</td>
</tr>
</tbody>
</table>

Table 8.5: Process no. 33.12.
Table 8.6: Process no. 33.13.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.12</td>
<td>33.13</td>
<td><a href="image">Diagram</a></td>
</tr>
<tr>
<td>Control fire plan data</td>
<td>Send the message repeat</td>
<td>Repeat!</td>
</tr>
<tr>
<td>OPS DO</td>
<td>OPS DO</td>
<td>&lt; repeat</td>
</tr>
</tbody>
</table>

Table 8.7: Process no. 33.14.

<table>
<thead>
<tr>
<th>Input from</th>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.12</td>
<td>33.14</td>
<td><a href="image">Diagram</a></td>
<td></td>
</tr>
<tr>
<td>Control fire plan data</td>
<td>Fire plan data</td>
<td><a href="image">Diagram</a></td>
<td></td>
</tr>
<tr>
<td>OPS DO</td>
<td>OPS DO</td>
<td>Fire plan data:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt; target number</td>
<td></td>
</tr>
<tr>
<td>&lt; OK</td>
<td>&lt; target number</td>
<td>&lt; from time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; from time</td>
<td>&lt; to time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; to time</td>
<td>&lt; east</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; east</td>
<td>&lt; north</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; north</td>
<td>&lt; registered by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; registered by</td>
<td>&lt; registered with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; registered with</td>
<td>&lt; ammo type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; ammo type</td>
<td>&lt; type of target</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; type of target</td>
<td>&lt; no. of batteries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt; no. of batteries</td>
<td>&lt; duration</td>
<td></td>
</tr>
<tr>
<td>Input from</td>
<td>PROCESS</td>
<td>Output to</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>$33.3$ Fire plan data</td>
<td>33.15</td>
<td>$33.1$ Fire plan</td>
<td></td>
</tr>
</tbody>
</table>

- target number
- from time
- to time
- east
- north
- registered by
- registered with
- ammo type
- type of target
- no. of batteries
- duration

< target number
< from time
< to time
< east
< north
< registered by
< registered with
< ammo type
< type of target
< no. of batteries
< duration
< no. of targets
< comments

Table 8.8: Process no. 33.15.

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.16</td>
<td>$33.3$ Fire plan data</td>
</tr>
</tbody>
</table>

- delete command

Table 8.9: Process no. 33.16.
Table 8.10: Process no. 33.17.

Table 8.11: Process no. 33.18.

8.1.9 RGPM models

Today there are non explicitly stated requirements or goals.

8.1.10 Goal Table

Today there are non explicitly stated requirements or goals.
8.2 Desired future state

8.2.1 Major structural changes

Figure 8.1 shows that the control span of the staff battery commander is big, too big. In addition, experiences show that the importance of logistics have been suppressed. Therefore, we have established a train battery which is part of the artillery battalion (Figure 8.23). We also believe that it will be more efficient and effective to include all the localization resources in a target acquisition battery as shown in Figure 8.23. This is also motivated by the goal "centralized coordination of firing resources and localization resources" (Figure 8.17).

8.2.2 Process overview model

The customer process intelligence shown in the current process overview model of the artillery (Figure 8.5) collect intelligence information from static forward observers and artillery locating radar’s and forwards the intelligence information to the intelligence center in the brigade HQ. Hostile artillery units, headquarters, logistic units, etc spotted by static forward observers and hostile artillery units detected by artillery locating radar’s are not part of present fire plans.

In the future we want hostile artillery units, headquarters, logistic units, etc. detected by static forward observers and artillery locating radar’s to be part of the fire plans. This is also motivated by the means indicated in the RGPM model shown in Figure 8.16. Fire plan data should also in the future be circulated to the brigade HQ, thus the intelligence center in the brigade HQ will still receive intelligence information from the static forward observers and the artillery locating radar’s.

The process overview model of the future is shown in Figure 8.13. The intelligence process is removed, but the intelligence center in the brigade HQ will still receive intelligence information as a spin-off of the fire planning process.
8.2.3 RGPM models

The customer process
Customers of the artillery, e.g. infantry and armor units, have the following high level requirements to the customer process, i.e. the firing process, its results and its equipment:

- flexibility
- range: up to 50 km
- at once
- when needed
- right place
- right effect.

In this case study we have presented RGPM models of the requirements flexibility (Figure 8.14) and when needed (Figure 8.15). All customers are symbolized by the infantry role indicator.
Figure 8.14: Customer requirement: flexibility.

Figure 8.15: Customer requirement: when needed.
Support processes
As shown in the process overview model (Figure 8.13) the firing process has four support processes: fire planning, logistics, maneuvering and establish technical baseline. In this example RGPM models are only made for the fire planning process.

The goal "fulfill 9 out of 10 request for fire" requires a very flexible and effective utilization of the firing resources, ie guns, and the localization resources, ie forward observers, static forward observers and artillery locating radar's.

The firing process should be smooth and fast, ie every possible target should be planned for, even opportunity targets.

Thus, the customer of the fire planning process, ie the firing process, has the following high level requirements to the support process "fire planning" and its results:

• every target planned for (Figure 8.16)
• effective utilization of firing and localization resources (Figure 8.17).

![Diagram](Figure 8.16: Customer requirement: every target planned for.)
8.2.4 Goal table

The goals assigned to achievers are summarized in Table 8.12.

<table>
<thead>
<tr>
<th>Achievers</th>
<th>Assigned goals</th>
</tr>
</thead>
</table>
| FCC       | - scheduled targets or opportunity targets, no difference  
                        - fulfill 9 out of 10 requests for fire  
                        - continuous fire planning  
                        - centralized coordination of the firing resources |

Table 8.12: Goal table.

8.2.5 Message sequence model

Figure 8.18 shows the message sequence model of the future fire planning process. The differences between Figure 8.12 (the current fire planning process) and Figure 8.18 are:

- Because of the reorganization mentioned in subsection 8.2.1 Major structural changes, the battery commanders are not involved in the fire planning process any more. In the future, all localization activity are directed by the target acquisition battery HQ.
We believe that it is more efficient to let forward observes transmit fire plan data directly to the artillery battalion instead of to the artillery battalion via the battery commanders and artillery batteries. Thus "Fire plan data" messages should in the future be transmitted directly to the artillery battalion HQ.

The control message "prepare ammo!" will be removed from the fire planning process of the future. The reason is that the gun group commanders and the officer responsible for the logistics in the artillery batteries are able to figure out on their own that they should prepare the ammunition. To be effective it is important to delegate not only tasks, but also responsibility and authority, and that is a question of trust.

To obtain centralized coordination of the firing resources target recording should take place in the FCC in the artillery battalion. Thus the "target recording!" orders should in the future be transmitted directly to the artillery battalion.

Since the there will be just one continuous evolving fire plan in the future, the messages "Fire plan" and "Fire plan ready!" are obsolete. In the future the message "Fire plan ready!" will be substituted by the message "Ready to fire!" while the message "Fire plan" will be substituted by circulated fire plan data.

The messages "Distribution of ammo!", "MET data!" and "Gunnery data calculations!" are removed. The explanations are given in the next subsection.

Figure 8.18: Message sequence model of the future fire planning process.
8.2.6 Process models

The figures in this subsection (Figure 8.19 to Figure 8.22) describes the desired future fire planning process at the battalion level. The fire planning process is decomposed at least until none of its sub-processes

- receive more than one message type from agents
- send more than one message type to agents
- receive messages from both other sub-processes and agents
- send messages to both other sub-processes and agents.

In addition to the already mentioned differences between the current fire planning process and the future fire planning process we have the following differences:

- Since the message "Target recording!" is transmitted to the artillery battalion HQ, the shelling data and gunnery data are calculated in the artillery battalion HQ. In addition, all the shelling data are circulated, not only the coordinates. The circulation of shelling data is motivated by the redundancy requirement discussed in section 2.2.
- The orders process shown in Figure 8.7 is not needed any more because the shelling data and gunnery data are calculated in the artillery battalion HQ.
- We have also transferred the ammunition control process shown in Figure 8.7 to the logistics process.

![Diagram of the fire planning process](image)

Figure 8.19: The fire planning process (first level).
Figure 8.20: The decomposition of the fire planning process (second level).

Note that process number 33.3 shown in Figure 8.20 is not decomposed any further.
Figure 8.22: The decomposition of the calculate shelling data and gunnery data process (third and final level).

8.2.7 Rule table

Rules for processes at the lowest level of decomposition are shown in Table 8.13.
<table>
<thead>
<tr>
<th>Processes</th>
<th>Rules</th>
</tr>
</thead>
</table>
| 33.3 | **When** the artillery operation order (Art.ops) is received  
**Then** trigger off the fire planning process  
and store the artillery operation order in the artillery operation order archive |
| 33.11 | **When** the fire plan data is received  
**Then** control the fire plan data against the safety data  
If the fire plan data is judged OK  
**Then** trigger off process no. 33.13 Store and circulate fire plan data  
Else trigger off process no. 33.12 Store and notify DO |
| 33.12 | **When** the fire plan data is not judged OK  
**Then** store the fire plan data in the store corrupted fire plan data  
and notify DO |
| 33.13 | **When** the fire plan data is judged OK  
**Then** circulate the fire plan data to all FAbt’s, the R-FAbn, the brigade and the division  
and store the fire plan data in the fire plan data store |
| 33.14 | **When** notified  
**Then** control the fire plan data against the safety data  
If the fire plan data is judged OK  
**Then** trigger off process no. 33.16 Store as fire plan data and circulate  
Else trigger off process no. 33.15 Send the message repeat |
| 33.15 | **When** the fire plan data is not judged OK  
**Then** send the message repeat to the unit that sent the fire plan data |
| 33.16 | **When** the fire plan data is judged OK  
**Then** circulate the fire plan data to all FAbt’s, the R-FAbn, the brigade and the division  
and store the fire plan data in the fire plan data store  
and delete the fire plan data from the store corrupted fire plan data |
| 33.17 | **When** the message ready to fire is received  
**Then**  
If two or more batteries and the reinforced FAbn have reported fire plan ready  
**Then** send the message ready to fire to the brigade and the division  
Else wait |
| 33.21 | **When** the command target recording is received  
**Then** calculate shelling data and control coordinates against the safety data  
If the coordinates are judged OK  
**Then** trigger off process no. 33.23 Store and circulate shelling data  
Else trigger off process no. 33.22 Store and notify DO |
When the coordinates are not judged OK
Then store the shelling date in the store corrupted shelling data
and notify DO

When the coordinate are judged OK
Then circulate the shelling data to all FAbt’s, the R-FAbn and the division
and store the shelling data in the shelling data store

When notified
Then control the coordinates against the safety data
If the coordinates are judged OK
Then trigger off process no. 33.26 Store as shelling data and circulate
Else trigger off process no. 33.25 Send the message repeat

When the shelling data is not judged OK
Then send the message repeat to the unit that sent the command target recording

When the coordinates are judged OK
Then circulate the shelling data to all FAbt’s, the R-FAbn and the division
and store the shelling data in the shelling data store
and delete the shelling data from the store corrupted shelling data

When time
Then calculate gunnery data

Table 8.13: Rule table, processes.

Rules for conditional messages are shown in Table 8.14.

<table>
<thead>
<tr>
<th>Messages</th>
<th>From</th>
<th>To</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready to fire!</td>
<td>Gun groups</td>
<td>FAbt</td>
<td>When the ammunition is prepared and guns in position</td>
</tr>
<tr>
<td>Ready to fire!</td>
<td>R-FAbn</td>
<td>FAbn</td>
<td>When two or more batteries have reported ready to fire</td>
</tr>
<tr>
<td>Ready to fire!</td>
<td>FAbt</td>
<td>FAbn</td>
<td>When the ammunition is prepared, and the technical baseline is settled, and two or more gun groups have reported ready to fire</td>
</tr>
<tr>
<td>Ready to fire!</td>
<td>FAbn</td>
<td>Brig</td>
<td>When two or more batteries and the reinforced FAbn have reported ready to fire</td>
</tr>
</tbody>
</table>

Table 8.14: Rule table, messages.
8.2.8 Organization structure models

Figure 8.23 shows the desired future organization structure of a field artillery battalion, Figure 8.24 shows the desired future organization structure of a field artillery battery, while Figure 8.25 shows the organization structure of a target acquisition battery.

Figure 8.23: Field artillery battalion.

Figure 8.24: Field artillery battery.
8.2.9 Role models

Figure 8.26 shows the centers and roles of the future battalion HQ. The most striking change is the introduction of the fire control center (FCC). This change is motivated by the strict requirements made to the firing process and the fire planning process, and the challenging goals assigned to the center in the artillery battalion responsible for the firing process and the fire planning process.

Figure 8.25: Target acquisition battery.

Figure 8.26: Battalion HQ.
Figure 8.27 shows the centers and roles of the future battery HQ. The most striking change is the introduction of the FCC. This change is motivated by the challenging goals assigned to the FCC in the artillery battalion. To be able to keep pace with the battalion a FCC is also needed by the batteries.

Figure 8.27: Battery HQ.

### 8.2.10 Responsibility table

Table 8.15 summarizes the responsibilities assigned to HQ, centers and roles.

<table>
<thead>
<tr>
<th>HQ</th>
<th>Centers</th>
<th>Roles</th>
<th>Responsible for process</th>
<th>P. no</th>
</tr>
</thead>
<tbody>
<tr>
<td>•</td>
<td>STAFF</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>OPS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FCC</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>FCO</td>
<td>Fire planning</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receive the artillery operation order</td>
<td></td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td>Ass FCO/DO</td>
<td>Fire plan data management</td>
<td></td>
<td>33.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control fire plan data</td>
<td></td>
<td>33.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Report ready to fire</td>
<td></td>
<td>33.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Calculate shelling data and gunnery data</td>
<td></td>
<td>33.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control coordinates</td>
<td></td>
<td>33.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Calculate gunnery data</td>
<td></td>
<td>33.27</td>
</tr>
<tr>
<td></td>
<td>HQ ass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WS</td>
<td>Control fire plan data</td>
<td></td>
<td>33.11</td>
</tr>
<tr>
<td>ADM</td>
<td>Store and notify DO</td>
<td>33.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Store and circulate fire plan data</td>
<td>33.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send the message repeat</td>
<td>33.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Store as fire plan data and circulate</td>
<td>33.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Calculate shelling data and control coordinates</td>
<td>33.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Store and notify DO</td>
<td>33.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Store and circulate shelling data</td>
<td>33.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send the message repeat</td>
<td>33.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Store as shelling data and circulate</td>
<td>33.26</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.15: Responsibility table.

### 8.2.11 Message content tables

Message content tables show the input and output of processes. Table 8.16 to Table 8.22 show the message content tables for the sub-processes of the fire plan data management process (based on Figure 8.21).

<table>
<thead>
<tr>
<th>Input from</th>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
</tr>
</tbody>
</table>

**Fire plan data:**
- target number
- from time
- to time
- east
- north
- registered by
- registered with
- ammo type
- type of target
- no. of batteries
- duration

- FLOT
- coordination lines
- position of friendly units in front of FLOT
- boundary lines

- target number
- from time
- to time
- east
- north
- registered by
- registered with
- ammo type
- type of target
- no. of batteries
- duration

Table 8.16: Process no. 33.11.
### Table 8.17: Process no. 33.12.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.11</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Control fire plan data</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>FCC _ WS</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>&lt; target number</td>
<td>&lt; target number</td>
<td>&lt; target number</td>
</tr>
<tr>
<td>&lt; from time</td>
<td>&lt; from time</td>
<td>&lt; from time</td>
</tr>
<tr>
<td>&lt; to time</td>
<td>&lt; to time</td>
<td>&lt; to time</td>
</tr>
<tr>
<td>&lt; east</td>
<td>&lt; east</td>
<td>&lt; east</td>
</tr>
<tr>
<td>&lt; north</td>
<td>&lt; north</td>
<td>&lt; north</td>
</tr>
<tr>
<td>registered by</td>
<td>registered by</td>
<td>registered by</td>
</tr>
<tr>
<td>registered with</td>
<td>registered with</td>
<td>registered with</td>
</tr>
<tr>
<td>ammo type</td>
<td>ammo type</td>
<td>ammo type</td>
</tr>
<tr>
<td>type of target</td>
<td>type of target</td>
<td>type of target</td>
</tr>
<tr>
<td>no. of batteries</td>
<td>no. of batteries</td>
<td>no. of batteries</td>
</tr>
<tr>
<td>duration</td>
<td>duration</td>
<td>duration</td>
</tr>
</tbody>
</table>

### Table 8.18: Process no. 33.13.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.11</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Control fire plan data</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>FCC _ WS</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>&lt; target number</td>
<td>Fire plan data:</td>
<td>&lt; target number</td>
<td>&lt; from number</td>
</tr>
<tr>
<td>&lt; from time</td>
<td></td>
<td>&lt; from time</td>
<td>&lt; to time</td>
</tr>
<tr>
<td>&lt; to time</td>
<td></td>
<td>&lt; to time</td>
<td>&lt; east</td>
</tr>
<tr>
<td>&lt; east</td>
<td></td>
<td>&lt; east</td>
<td>&lt; north</td>
</tr>
<tr>
<td>&lt; north</td>
<td></td>
<td>&lt; north</td>
<td>registered by</td>
</tr>
<tr>
<td>registered by</td>
<td></td>
<td>registered by</td>
<td>registered with</td>
</tr>
<tr>
<td>registered with</td>
<td></td>
<td>registered with</td>
<td>ammo type</td>
</tr>
<tr>
<td>ammo type</td>
<td></td>
<td>type of target</td>
<td>type of target</td>
</tr>
<tr>
<td>type of target</td>
<td></td>
<td>type of target</td>
<td>no. of batteries</td>
</tr>
<tr>
<td>no. of batteries</td>
<td></td>
<td>no. of batteries</td>
<td>duration</td>
</tr>
<tr>
<td>duration</td>
<td></td>
<td>duration</td>
<td></td>
</tr>
<tr>
<td>Input from</td>
<td>PROCESS</td>
<td>Output to</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>[33.4] Safety data</td>
<td>Control fire plan data</td>
<td>Send the message repeat</td>
<td></td>
</tr>
<tr>
<td>[33.5] Corrupted fire plan data</td>
<td>[FCC ___ DO]</td>
<td>[FCC ___ WS]</td>
<td></td>
</tr>
<tr>
<td>&lt; target number</td>
<td>&lt; repeat</td>
<td>&lt; target number</td>
<td></td>
</tr>
<tr>
<td>&lt; from time</td>
<td>&lt; repeat</td>
<td>&lt; from time</td>
<td></td>
</tr>
<tr>
<td>&lt; to time</td>
<td>&lt; repeat</td>
<td>&lt; to time</td>
<td></td>
</tr>
<tr>
<td>&lt; east</td>
<td>&lt; repeat</td>
<td>&lt; east</td>
<td></td>
</tr>
<tr>
<td>&lt; north</td>
<td>&lt; repeat</td>
<td>&lt; north</td>
<td></td>
</tr>
<tr>
<td>&lt; registered by</td>
<td>&lt; repeat</td>
<td>&lt; registered by</td>
<td></td>
</tr>
<tr>
<td>&lt; registered with</td>
<td>&lt; repeat</td>
<td>&lt; registered with</td>
<td></td>
</tr>
<tr>
<td>&lt; ammo type</td>
<td>&lt; repeat</td>
<td>&lt; ammo type</td>
<td></td>
</tr>
<tr>
<td>&lt; type of target</td>
<td>&lt; repeat</td>
<td>&lt; type of target</td>
<td></td>
</tr>
<tr>
<td>&lt; no. of batteries</td>
<td>&lt; repeat</td>
<td>&lt; no. of batteries</td>
<td></td>
</tr>
<tr>
<td>&lt; duration</td>
<td>&lt; repeat</td>
<td>&lt; duration</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.19: Process no. 33.14.

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>[33.14] [DO]</td>
<td>[33.15] [DO]</td>
<td>[DO]</td>
</tr>
<tr>
<td>Control fire plan data</td>
<td>Send the message repeat</td>
<td></td>
</tr>
<tr>
<td>[FCC ___ DO]</td>
<td>[FCC ___ WS]</td>
<td></td>
</tr>
<tr>
<td>&lt; repeat</td>
<td>&lt; repeat</td>
<td>&lt; repeat</td>
</tr>
</tbody>
</table>

Table 8.20: Process no. 33.15.
**Table 8.21: Process no. 33.16.**

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; target number</td>
<td>Fire plan data:</td>
<td>&lt; target number</td>
</tr>
<tr>
<td>&lt; from time</td>
<td>&lt; target number</td>
<td>&lt; from time</td>
</tr>
<tr>
<td>&lt; to time</td>
<td>&lt; from time</td>
<td>&lt; to time</td>
</tr>
<tr>
<td>&lt; east</td>
<td>&lt; to time</td>
<td>&lt; east</td>
</tr>
<tr>
<td>&lt; north</td>
<td>&lt; east</td>
<td>&lt; north</td>
</tr>
<tr>
<td>&lt; registered by</td>
<td>&lt; north</td>
<td>&lt; registered by</td>
</tr>
<tr>
<td>&lt; registered with</td>
<td>&lt; registered with</td>
<td>&lt; registered with</td>
</tr>
<tr>
<td>&lt; ammo type</td>
<td>&lt; type of target</td>
<td>&lt; ammo type</td>
</tr>
<tr>
<td>&lt; type of target</td>
<td>&lt; type of target</td>
<td>&lt; type of target</td>
</tr>
<tr>
<td>&lt; no. of batteries</td>
<td>&lt; no. of batteries</td>
<td>&lt; no. of batteries</td>
</tr>
<tr>
<td>&lt; duration</td>
<td>&lt; duration</td>
<td>&lt; duration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output to</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupted fire plan data</td>
<td>Corrupted fire plan data</td>
</tr>
</tbody>
</table>

* delete command

**Table 8.22: Process no. 33.17.**

<table>
<thead>
<tr>
<th>Input from</th>
<th>PROCESS</th>
<th>Output to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready to fire:</td>
<td>Report ready to fire</td>
<td>Ready to fire:</td>
</tr>
<tr>
<td>&lt; ready to fire</td>
<td></td>
<td>&lt; ready to fire</td>
</tr>
</tbody>
</table>
9 Evaluation

Our domain language is based on practical usefulness, it is developed in close cooperation with CCIS domain experts, it is domain appropriate, it works, and some of the sub-languages are already in use. The main weakness so far is the lack of tool support, ie a user-friendly tool with proper drawing editors and a useful repository (see section 9.2 Tool support).

It is important to use our domain language in a cooperative oriented modeling approach, eg participatory design or JAD. IS engineers will not be able to understand a domain by just using a domain language, or whatever the language, but concepts like participatory design and JAD will neither come into its full rights without using a language that are fully understood by the users, eg a domain language.

"A pre-requisite for obtaining cooperation is that the participants share a language by which they can communicate." [Sølvberg96]

Thus, cooperative oriented approaches where users and IS engineers are modeling and working together as a team are the key to proper communication and common understanding between the two.

Our domain language is not powerful enough to express anything in the domain, but we don’t think it is appropriate to develop a domain modeling language that is powerful enough to express anything in the domain because such a language will not be practical. We believe the best solution is to have a domain language that is able to express the most important phenomena in the domain and to fill in with natural language or other informal languages where it is appropriate.

We have claimed that:

Small IS models are mostly understandable for the users, but as the IS models grow they tend to become very difficult to understand and therefore to verify and validate.

Surely, the same problems apply to models made by domain languages, even for those that are extremely well known in a particular domain. However, we believe that use of known terminology and graphical symbols in models (ie domain models) will raise the threshold of what the users find difficult to understand. It is also our intention, as stated above, to use the domain language in a cooperative oriented modeling approach where users and IS engineers are modeling and working together as a team, and we all know
that it is much easier to understand complex models when we have been involved in creating them. Even IS engineers find it difficult to understand complex IS models that they have not been involved in creating themselves, even if the language used is familiar to them.

9.1 Additional usage

The main purpose of domain languages is to use them in the domain redesign stage of the information system life cycle model as described in chapter 4. However, domain languages could be used in additional ways, e.g., they could be used to make descriptive domain models as an aid to document how the domain works. The descriptive domain models could then be used in connection with education.

Our domain language could also be of value in connection with military operations, like the operational modeling language presented in [APP-6]. The domain language could be used as a tool to redesign the domain after losses, reinforcement, etc., i.e., it could be used to update message sequence models (i.e., models showing who is sending which messages to whom), process models (i.e., models showing tasks and responsibilities), role models (i.e., models showing the organization of headquarters), and organization structure models. Thus, domain models could be used to graphically depict the status of the domain and at the same time graphically depict how the domain should work to keep it up.

9.2 Tool support

We have implemented a prototype tool [Olsen95] to support our domain language. Four out of six drawing editors have been implemented. Each drawing editor supports one of the six sub-languages. The four implemented sub-languages have not yet been tightly integrated since the development environment lacks a suitable repository solution; however, an example of how the integration can be done is implemented. A suitable repository is currently under development.

The drawing editors are quite straightforward to implement. The challenge is to develop a repository that:

- makes it possible to tightly integrate the sub-languages into a compound language as shown by the metamodel (Figure 7.58 and Figure 7.59)
- is able to manage versioning, multiple users and merging
- support views, i.e., it should be possible to present different views of the same model. For example, to facilitate the readability of organization structure models, we strongly recommend to view maximum three levels of an organization structure model simultaneously.

A complete tool should on request control if there are inconsistencies between the
models created by the different editors. It should also on request propagate changes made to a model to other models influenced by the changes.

The models created by the editors are independent of each other. Inter-model relations should be handled by a repository interface layer (RIL) that isolates the local repository from the editor repositories (ER), as shown in Figure 9.1.

![Diagram](image)

Figure 9.1: Repository interface layer.

A change in a model is on request reported to the repository interface layer. The repository interface layer then determine if the change could cause any changes in other models, and sends messages accordingly to update these models. The information in the local repository are also changed by this operation. Thus, the repository interface layer must "know" the metamodel of the whole domain language, while each editor repository is only concerned with the sub-language it supports and its metamodel.

The current version of the tool does not maintain any inter-model relations, but some relations are built temporarily when a message sequence model is created from a process model.

### 9.3 Connections between our domain language and domain independent modeling languages

There are hundreds of different domain independent modeling languages [Siau96], but it is not our intention to assess how all of them could be connected to our domain language. We will concentrate on two of the most commonly used modeling
approaches, ie SA (structured analysis) and OOA (object oriented analysis).

### 9.3.1 Object oriented analysis

Our domain language fits the OOA approach rather good. Organization structure models and role models could be regarded as object class models, the message sequence models show the message passing between object classes, while the process models could be regarded as operations performed by object classes.

### 9.3.2 Structured analysis

Normally, two languages are included in SA: DFD (data flow diagram) and ER (entity-relationship).

Our process language subsumes the traditional data flow diagram (DFD) language. It is therefore relatively simple to transform our process models, or part of them, to traditional DFD models if necessary. In Figure 9.2 the process model of the fire planning process, third battalion level (see Figure 7.43) is transformed to a traditional DFD.

![Diagram](image)

Figure 9.2: DFD of the fire planning process, third bn level.
In data flow diagrams it is not possible to distinguish between computerized and manual processes, neither is it possible to distinguish between data stores and paper stores. DFDs should therefore only be used to model computerized parts as shown in Figure 9.3.

![Diagram of DFD](image)

Figure 9.3: DFD of the computerized parts of the fire planning process, third bn level.

Data modeling in terms of ER modeling or related techniques is not part of our domain language. However, information required by processes and information sent between units/HQ are analyzed and specified in details. The basis for data modeling, eg ER modeling, is therefore in place.
10 Conclusion

This chapter concludes the thesis. The major results and achievements in the thesis are summarized in section 10.1, while the directions for further work are outlined in section 10.2.

10.1 Results

The goal stated in chapter 1 is met, i.e. we have developed a domain language for military command, control and information systems that is:

- based on the users’ professional language
- developed on the behalf of the users (i.e. domain experts)
- closer related to traditional IS modeling languages than natural languages are.

The main contribution of this thesis is the development of the domain language. In more details the contributions are:

- the idea of using domain specific constructs in a modeling language.
- the combination and integration of sub-languages in such a way that the compound language is especially suited for describing a particular domain.
- the process overview sub-language.
- the RGPM sub-language.
- the process sub-language. It is based on DFD [Gane79], but major adjustments are done. In addition appropriate domain specific constructs are added to extend its expressive power.
- the message sequence sub-language. It is inspired by CCITT [CCITT120] and the scenario language in OOram [Reenskaug96].
- the role sub-language. It is based on the traditional organization chart language, but this way of using it is the author's idea.
- the organization structure sub-language. It is based on the traditional organization chart language used by the Norwegian Army, but the addition of cardinalities and the standardized way to draw the organization charts are the author's idea. The idea of integrating the organization structure language with other modeling language is the author's too.
- the domain oriented tables.
10.2 Further work

Several areas for further work can be envisaged. Four of them are introduced in the following subsections:

- tool support
- domain language environment
- transformation of domain models to IS models
- domain languages used in a domain analysis approach.

10.2.1 Tool support

A user-friendly tool with proper drawing editors and a suitable repository has to be developed before our domain language may gain extensive use (see section 9.2 Tool support).

The biggest challenge is to develop a repository that:

- makes it possible to tightly integrate the sub-languages into a compound language as shown by the metamodel (Figure 7.58 and Figure 7.59)
- is able to manage versioning, multiple users and merging
- support views, ie it should be possible to present different views of the same model.

10.2.2 Domain language environment

Because there are many different domains, it might be economical to develop a tool suitable for building domain languages, ie a domain language environment. We believe that the core of a domain language environment must at least consists of five parts, as illustrated in Figure 10.1:

- a language editor for building languages and configuration of drawing editors
- a metamodeling language
- a generic drawing editor that easily adapts to new languages
- a generic repository interface layer (RIL) that easily adapts to new metamodels
- a generic repository that easily adapts to new concepts and relations.

Again we believe the biggest challenge to be the repository because it is the repository that dictates the flexibility of the domain language environment and thus the environments ability to build suitable domain languages. We believe that a fixed repository structure will impose to many restrictions on the domain language environment and thus reduce the environments ability to build suitable domain languages.
10.2.3 Transformation of domain models to IS models

It is important to maintain vertical consistency between semantically equivalent models and other design descriptions created at different development stages. This could be done manually, but tool support for automatically transformation of domain models to IS models to maintain vertical consistency would clearly strengthen our proposed domain language.

Suggestions for further work are:

• identify and select suitable IS models
• develop methods for transformation of domain models to selected IS models
• develop tool support for automatically transformation of domain models to selected IS models.

10.2.4 Domain languages used in domain analysis approaches

Domain languages should be used to support domain analysis/engineering approaches (see chapter 4) focusing on reuse of domain models and domain knowledge, ie domain languages should be used in the activity of identifying and documenting common and variable parts of related domains.

Research on how to use domain languages in a reuse oriented context should be conducted.
Bibliography


[Batory95] Batory, D., Coglianese, L., Goodwin, M., Shafer, S. Creating Reference


Vol. 36, no. 4, June 1993.


Abbreviations

ADM  Administration center
Ammo  Ammunition
Art.ops  Artillery operation order
Ass  Assistant
BC  Battery commander
Bn  Battalion
BnC  Battalion commander
BO  Battery officer
Brig  Brigade
Btt  Battery
C&CIS  Command and control information system
C2RA  Command and control requirements analysis
CCIS  Command, control and information system
CIS  Computerized information system
Conv  Conventional
CP  Command post
DFD  Data flow diagram
Div  Division
DO  Duty officer
EIS  Executive information system
ER  Entity-relationship
FA  Field artillery
FAbn  Field artillery battalion
FAbtt  Field artillery battery
FCC  Fire control center
FCO  Fire control officer
FLOT  Forward line of own troops
FO  Forward observer
FSCC  Fire support co-ordination center
HQ  Headquarters
HQ off  Headquarters officer
IFIP  International Federation for Information Processing
Infbn  Infantry battalion
Intrep  Intelligence report
IS  Information system
IT Information technology
JAD Joint Application Development
LANDOC Land operations command
Loc. orders Localization orders
MET Meteorological
NCF NATO Composite Force
NORCCIS II Norwegian Command, Control and Information System, phase II
NTNU The Norwegian University of Science and Technology
OMT Object Modeling Technique
OOA Object oriented analysis
OOram Object Oriented role analysis and modeling
OPS Operations center
OPS off Operations officer, battery level
PD Participatory design
R Reinforced
Rbn Reinforced battalion
Rec off Reconnaissance officer
RGPM Requirement, goal, problem and mean
RO Radio operator
S-1 Personnel officer
S-2 Intelligence officer, battalion level
S-3 Operations officer, battalion level
S-4 Administration officer
SA Structured analysis
Sit Situation
Sitrep Situation report
WS Work station
XO Executive officer, ie second in command