

HOVEDOPPGAVE

Kandidatens navn: Kenneth Aron Bjørkhaugen

Fag: Systemutvikling

Oppgavens tittel (norsk): MACCIS – En arkitektur evaluering

Oppgavens tittel (engelsk): MACCIS – An architectural evaluation

Oppgavens tekst:

MACCIS II (Minimal Architecture for C2IS in the Norwegian Army) is an architecture framework for CCIS (Command and Control Information System), made for the Norwegian Army. CCIS is an information system for planning, message handling and situational awareness at tactical level in the Army. This framework is developed by SINTEF Telecom and Informatics.

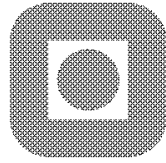
The goal of this project is to evaluate MACCIS II with emphasis on connection, cost efficiency and system development.

Connection is how the framework handles traceability and interoperability. Traceability reflects on the architecture and interoperability reflects on the different actors in the system.

Cost efficiency is a measure on how well the framework is doing what it is supposed to do, and also how well it is excluding what it is not supposed to do.

System development is concerned with how the framework is handling the system evolution, i.e. the development process.

NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
FAKULTET FOR FYSIKK , INFORMATIKK OG MATEMATIKK



Oppgaven gitt: 20. januar 2002
Besvarelsen leveres innen: 17. juni 2002
Besvarelsen levert: 17. juni 2002
Utført ved: På Jørstadmoen ved SBUKS – Utdannings- og
Kompetansesenter for Hærens Samband
Veileder: Siv. Ing. og Major Roger Johnsen

Trondheim,

Faglærer

ABSTRACT

This project evaluates an architectural framework developed primarily to describe the architecture of an information system made for the Norwegian Defence.

Today the Norwegian Defence consists of many isolated information system, many in which are built upon different platforms and with a vast variation of interfaces. This makes an integration of these systems very difficult, if not impossible. It definitely will make the integration very expensive, both in time and money.

Because of this a small group is working on an architectural framework, which is supposed to be implemented in the entire Defence. This framework is meant to describe every architecture in the same way and hence make integrations and new development much more effective and cost reduced.

The initial objectives for this project have been threefold. The first objective has been to look at the connection between the architectural framework and the information system. And by that see how the framework handles traceability and interoperability.

The second objective has been to see how cost effective the framework is, i.e. weather or not framework is doing what it is supposed to do, and also how well it is excluding what it is not supposed to do.

The third and last objective has been to look at the system development. System development is concerned with how the framework is handling the system evolution, i.e. the development process.

PREFACE

This report contains the documentation of the MACCIS II evaluation project with background in the Master degree thesis at the Department of Computer and Information Science, at the Norwegian University of Science and Technology during spring 2002. This thesis is part of the 10th and last semester in the Master of Science degree. The student responsible for this report is Kenneth Aron Bjørkhaugen.

This report has been written under supervision of Graduate engineer Roger Johnsen. The report will be offered to the architecture program, which is ongoing work in the Norwegian Defence.

I would like to thank my teaching supervisor Graduate engineer Roger Johnsen for the problem description and for finding time to guide me whenever I wanted guidance. I also would like to thank the rest of the employees at the Security Department at Army Signals Education and Training Center.

Jørstadmoen, June 14th 2002

Kenneth Aron Bjørkhaugen

TABLE OF CONTENTS

Abstract.....	i
Preface.....	ii
Table of Contents.....	iii
List of Tables.....	v
Table of Figures.....	vi
1. Introduction.....	1
1.1. Motivation.....	2
1.2. Project Context.....	3
1.3. Problem Definition.....	3
1.4. Document Structure.....	4
2. Prestudy.....	5
2.1. MACCIS II.....	6
2.1.1. System Architecture Model.....	7
2.1.2. Model Assets.....	10
2.1.3. System Scope.....	10
2.1.4. Component Technology.....	13
2.1.5. Modeling Language.....	14
2.1.6. Baseline Development Process.....	14
2.2. CCIS – System Overview.....	17
2.2.1. Functionality and Extent.....	17
2.2.2. Communication Infrastructure.....	18
2.2.3. Information Infrastructure.....	18
2.2.4. User Environment.....	19
2.3. DOLCCIS.....	20
2.4. Definition of Concepts.....	21
2.4.1. Network Centric Defence.....	21
2.4.2. Rational Unified Process.....	21
2.4.3. Reference Model for Open Distributed Processing.....	23
3. Discussions.....	27
3.1. Connectivity.....	28
3.1.1. Interoperability.....	28
3.1.2. Traceability.....	29
3.2. Cost Effectivity.....	30
3.2.1. Number of Models.....	30

3.2.2. Application of Models on Real-World Problems.....	31
3.3. System Development.....	32
3.3.1. Development Tool.....	32
3.4. Security.....	35
3.5. Parallel Development.....	36
4. Conclusion and Further Work.....	37
4.1. Conclusion.....	38
4.2. Further Work.....	39
4.3. Personal Experience and Opinion.....	40
5. Glossary.....	41
5.1. Military Specific Acronyms.....	42
5.2. General Acronyms.....	43
Bibliography.....	44

LIST OF TABLES

Table 3-1 Security functions in RM-ODP..... 35

TABLE OF FIGURES

Figure 2-1: MACCIS II Framework.....	6
Figure 2-2: Model Hierarchy - Generic View.....	7
Figure 2-3: Model overview.....	8
Figure 2-4: Model Hierarchy - MACCIS View.....	9
Figure 2-5: Systems Hierarchy - Generic View.....	10
Figure 2-6: System Model Hierarchy - Generic View.....	11
Figure 2-7: System Model Hierarchy - MACCIS II View.....	12
Figure 2-8: Software Component Reference Architecture - MACCIS II View..	13
Figure 2-9: Development Process - MACCIS View.....	15
Figure 2-10: Process Activities.....	16
Figure 2-11: Platform-, network- and information centric defence.....	21
Figure 2-12: RUP development phases.....	22
Figure 2-13: Layering and OSI.....	24
Figure 2-14: Viewpoints and RM-ODP.....	24

1 INTRODUCTION

1.1. MOTIVATION	2
1.2. PROJECT CONTEXT	3
1.3. PROBLEM DEFINITION	3
1.4. DOCUMENT STRUCTURE	4

1. Introduction

These sections outline the motivation and the context for this thesis, as well as the problem definition.

1.1. Motivation

Today the Norwegian Defence consists of many isolated information system, many in which are built upon different platforms and with a vast variation of interfaces. This makes an integration of these systems very difficult, if not impossible. It definitely will make the integration very expensive, both in time and money.

Because of this a small group is working on an architectural framework, which is supposed to be implemented in the entire Defence. This framework is meant to describe every architecture in the same way and hence make integrations and new development much more effective and cost reduced.

An architectural framework defines how architectures should be described. In addition to identifying architectural artifacts, a framework should also contain guidelines for how to produce the architectural artifacts.

MACCIS II [1 and 2] (Minimal Architecture for Command and Control Information System) is such an architectural framework, but targeted only towards software systems. This framework is meant to be a common reference to all architecture descriptions in the Norwegian Army. With such a framework applied on all the software systems in the Army, it will be much more effective and cost reductive to maintain existing systems, to add new systems and to replace one or more systems with another. That is because a common framework will make the same representation (structure) to all architectures, hence it will be easier to see the different interfaces which has to interoperate to function correctly.

CCIS (Command and Control Information System) is a relatively new information system for the Norwegian Army, and shall support operations within fire support, command and control and logistics support. That is, a system for planning, message handling and situational awareness at tactical level in the Army. The work on CCIS started in 1998 and is implemented during spring 2002. The so-called Block 1 (first deliverable of the CCIS) has been tested and implemented since December 2001.

The first deliverable of CCIS – *CCIS Block 1* – will hereby only be called CCIS.

The need for an architectural framework arose in parallel with the development of the CCIS. This gave birth to the MACCIS and later the revised version MACCIS II, which was finished December 2001.

SINTEF Telecom and Informatics have developed the MACCIS framework after commission from The Norwegian Army Material Command (FLO/LAND), called HFK before January 2002.

This graduate thesis will evaluate MACCIS II with some objectives taken into consideration. These objectives are outlined in section 1.3.

1.2. Project context

This work has been carried out based on an idea developed by Major Roger Johnsen, the commanding officer at the Security Department at the Army Signals Education and Training Center. Johnsen came up with some objectives, which became the foundation of the problem definition. These objectives are outlined in section 1.3.

The work on MACCIS started in 1998, almost concurrent with the work on CCIS. The overall goal of MACCIS is to provide an architectural framework to describe the architecture of CCIS. MACCIS has been revised into a new version called MACCIS II, which was finished December 2001.

The initial objectives of this thesis have been three-fold. The first objective has been to look at the connectivity, i.e. interoperability and traceability. The second objective has been to evaluate MACCIS II upon cost effectivity. The third, and last objective has been to look at the system development process in MACCIS II.

1.3. Problem definition

As mentioned above the problem definition is based upon three objectives.

The first objective is dealing with connectivity, i.e. interoperability and traceability. Interoperability is concerned with how well architectures described within the scope of MACCIS II is interoperable with other architectures both within The Norwegian Defence and without, i.e. NATO. The traceability is concerned with three aspects. The first is to look at traceability among the models developed within the scope of MACCIS II. The second aspect looks at traceability between different versions of the same model or set of models. The last aspect is evaluating the traceability among corresponding models in other sub-activities.

The second objective is concerned with cost effectivity, i.e. whether or not the architectural framework is fulfilling its initial goals. Two aspects are mentioned. One is the number of models. As chapter 2 outlines, MACCIS II consist of a large number of models, which is discussed in section 3.2.1. The second aspect is concerned with how the different models shall be used on real-world problems. In order to get an uniform application of models the framework should provide practical examples and comments on essential parts.

The third, and last objective, is concerned with the development process used in MACCIS II and some significant factors that should be used in the evaluation of a development tool meant used for MACCIS II.

1.4. Document structure

This project starts out with an abstract and a preface. Followed by chapter 1, which is concerned with motivation, project context and problem definition.

Chapter 2 presents different systems, technologies and concepts, e.g. MACCIS II, CCIS and RM-ODP.

Chapter 3 discusses the essence of this project, i.e. how MACCIS II is related to the three aspects outlined in section 1.3.

Chapter 4 contains a conclusion and outlines suggestions to further work.

Chapter 5 is a collection of all the acronyms found in this project.

The last chapter offers a bibliography.

2 PRESTUDY

2.1.	MACCIS II.....	6
2.1.1.	SYSTEM ARCHITECTURE MODEL.....	7
2.1.2.	MODEL ASSETS	10
2.1.3.	SYSTEM SCOPE.....	10
2.1.4.	COMPONENT TECHNOLOGY.....	13
2.1.5.	MODELLING LANGUAGE.....	14
2.1.6.	BASELINE DEVELOPMENT PROCESS	15
2.2.	CCIS – SYSTEM OVERVIEW	17
2.2.1.	FUNCTIONALITY AND EXTENT	17
2.2.2.	COMMUNICATION INFRASTRUCTURE.....	18
2.2.3.	INFORMATION INFRASTRUCTURE.....	18
2.2.4.	USER ENVIRONMENT	19
2.3.	DOLCCIS.....	20
2.4.	DEFINITION OF CONCEPTS	21
2.4.1.	NETWORK CENTRIC DEFENCE	21
2.4.2.	RATIONAL UNIFIED PROCESS	21
2.4.3.	REFERENCE MODEL FOR OPEN DISTRIBUTED PROCESSING	23

2. Prestudy

These sections give an introduction to the architectural framework MACCIS II and to the CCIS.

MACCIS II is an architectural framework based upon The Reference Model for Open Distributed Processing (RM-ODP) [3] and targeted towards software systems. RM-ODP is a joint standardization activity by both ISO and ITU-T. RM-ODP is further described in section 2.4.3. A framework like this defines how architectures should be described. In addition to identifying architectural artifacts, MACCIS II also contains guidelines for how to produce the architectural artifacts.

2.1. MACCIS II

MACCIS II defines a generic framework for system architecture that can be applied in any domain. Figure shows both the generic framework and a C2IS specific version of the framework.

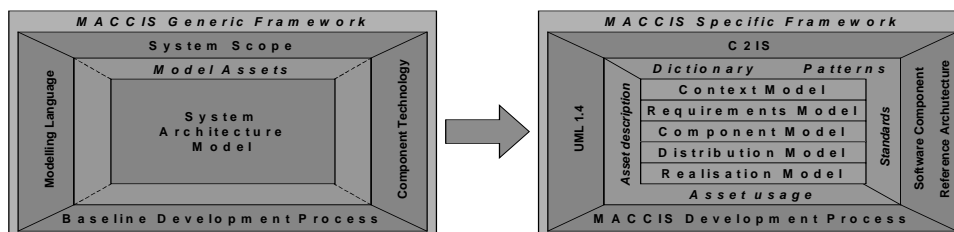


Figure 2-1: MACCIS II Framework

The generic framework focuses on system architecture models. The system architecture models are specified in a modelling language following a baseline development process. The system has a domain as specified in its scope and a set of target deployment technologies. In order to support system architecture modelling, a number of model assets are available, adapted to the language, process, scope and technologies. The specific architecture shows the refined framework for the C2IS scope. One advantage of exposing the generic framework is the ability to more cleanly adapt the specific refinement according to external changes.

2.1.1. System Architecture Model

Figure depicts the generic model hierarchy that is used as a foundation in the MACCIS II framework.

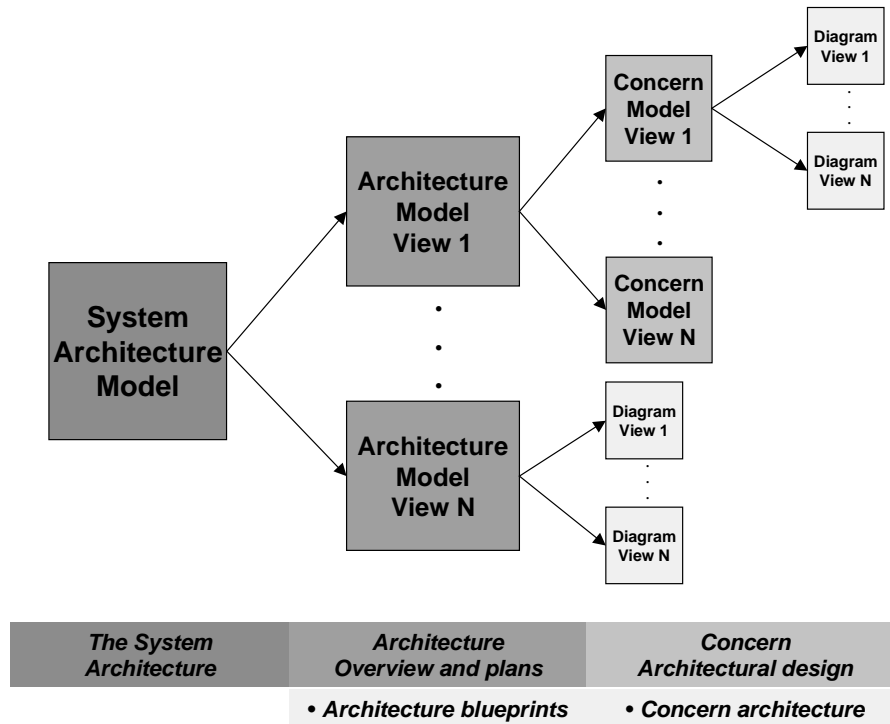


Figure 2-2: Model Hierarchy - Generic View

The system architecture model is the architectural description of what is regarded as the system. The system architecture model can be divided into a number of views, architecture model views, which describe important aspects of the system under consideration. Figure below shows the five views offered by MACCIS II. Each view has a set of essential models and a set of supporting models. The essential models is said to be necessary and the supporting models is said to be optional. The supporting models can be derived from the essential models.

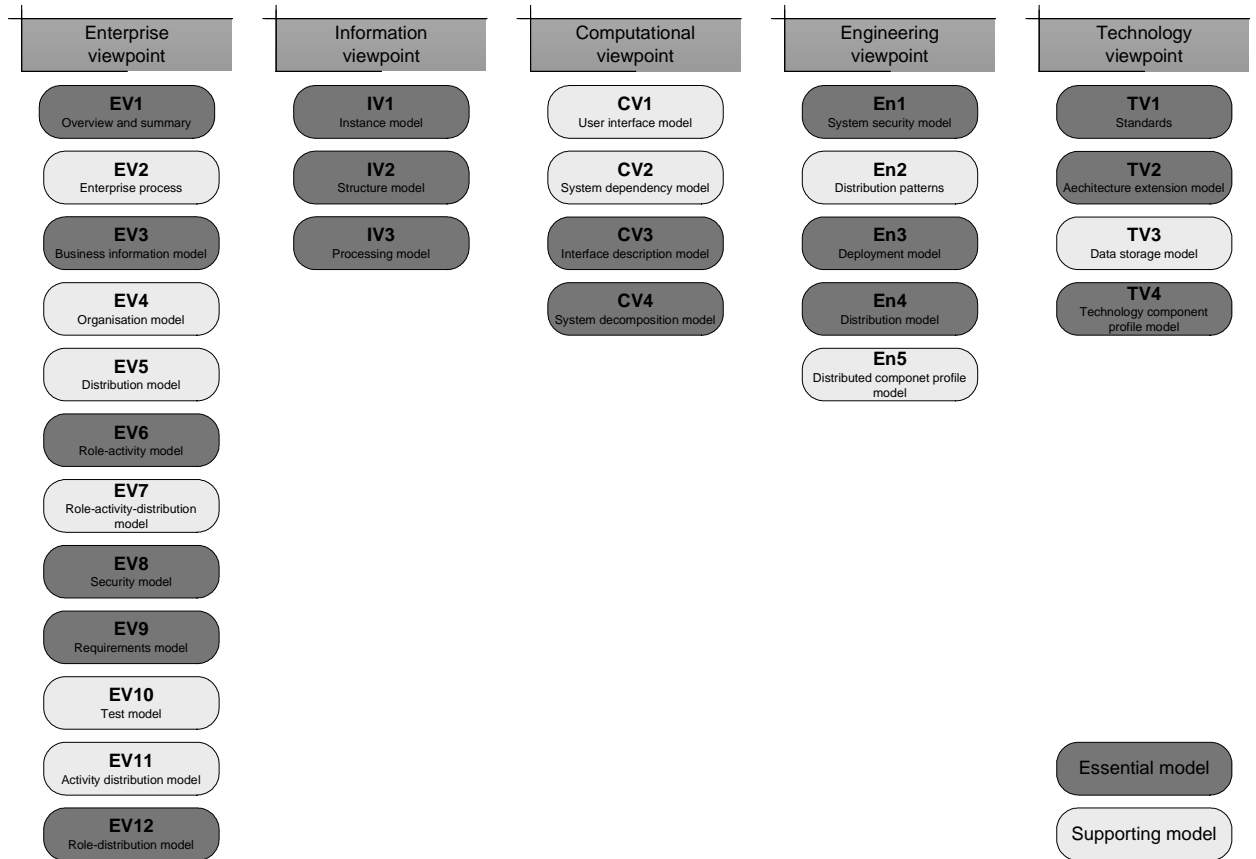


Figure 2-3: Model overview

Within each such viewpoint that defines the views, the overall structure and behavior in terms of system concerns are described. The main concern is the functionality of the system. A system concern is fully described as a set of model views – one for each architectural view defined.

In addition to the architecture model views and the concern model views, one can also create diagram views that describe selected parts of an architecture model view and/or concern model view at the necessary detail level required by the audience. Diagrams are used as a visual means of developing the architecture models required, but this does not suggest that all diagrams created during system development are useful to document the system architecture. The MACCIS II development process suggests a set of useful diagrams depicting the architecture blueprints of the system and the architecture of the concerns.

MACCIS defines system architecture in terms of five architecture model views:

- **Context Model:** The purpose of the context model is to describe the operation of the target system within a given context. This model also defines the dependencies of the specified elements of the target system to the specified elements of the context system.
- **Requirements Model:** The purpose of this model is to capture the requirements of the target system.
- **Component Model:** The purpose of this model is to describe the services, information, components and interactions of the target system.
- **Distribution Model:** The purpose of this model is to describe the distribution concerns of the target system.
- **Realisation Model:** The purpose of the realisation model is to describe the realisation of the target system in terms of technology and realized subsystems.

Figure depicts a matrix showing the architectural views and the concerns views for the specific model hierarchy addressed by the MACCIS II framework. The figure illustrates the use of model views for the component architecture view and the security concern view. The component architecture model describes the subsystem decomposition, information and interaction for all of the specified concerns. The security component model describes the security concern in the component architecture view. The security architecture model consists of a set of security model views that together describes the complete security architecture.

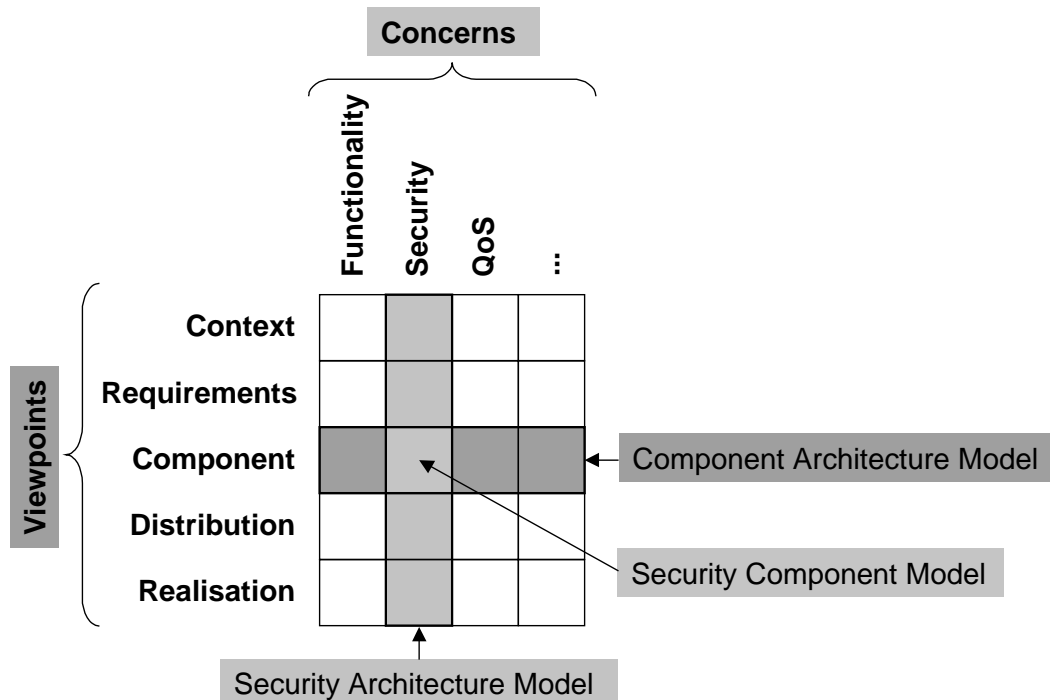


Figure 2-4: Model Hierarchy - MACCIS View

The set of concerns addressed is dependent of the target system and can also change over time. However, function is in most cases the most important concern since it is a

prerequisite for many other concerns, i.e. one must know what the system does before one is interested in how well it does it (Quality of Service - QoS) or the impact a malicious environment may have on it (security).

2.1.2. Model Assets

Assets are sources of information that can be used when developing the architecture models. Examples of assets that are available are:

- **Dictionary:** A dictionary is a reference list of concepts important to a particular model aspect or concern along with discussion and/or definition of their meanings and applications.
- **Standards:** A standard is a formalized model or example developed by a standardization organization or established by general consent. When implementing a C2IS, a set of various standards will probably be used, and should thus be referenced or documented.
- **Patterns:** A pattern is a description of a recurring, well-known problem and a suggested solution. Patterns are identified and can be used on many system levels.

2.1.3. System Scope

When specifying the architecture of a system, one needs to address the context for the system. Furthermore, one must address the realization of the target system in terms of its subsystems. The dependencies between these considerations are shown in Figure , which depicts a generic hierarchy and dependency model for systems engineering:

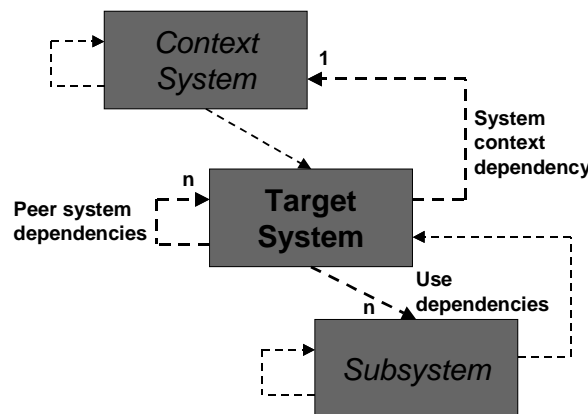


Figure 2-5: Systems Hierarchy - Generic View

The target system represents the system to be developed. The context system represents the higher-level system that the target system should support or be integrated with. This is represented by the “system context dependency”. The subsystem represents component building blocks that are specified or used when developing the target system. This is represented by the “use dependencies”. The target system may also have dependencies to other peer systems at the same level. This is represented by the “peer system dependencies”.

The model described above is generic from the target system’s perspective, and it can be recursively applied to context systems or subsystems as described above. Thus, the model can be used to describe infinite numbers of “systems” at various levels. The MACCIS II framework specifies how each of these system levels can be described using a set of UML models. The framework has been developed with the intent of minimizing the number of required models.

Figure depicts the generic scope for the system and model view that is used by the MACCIS framework:

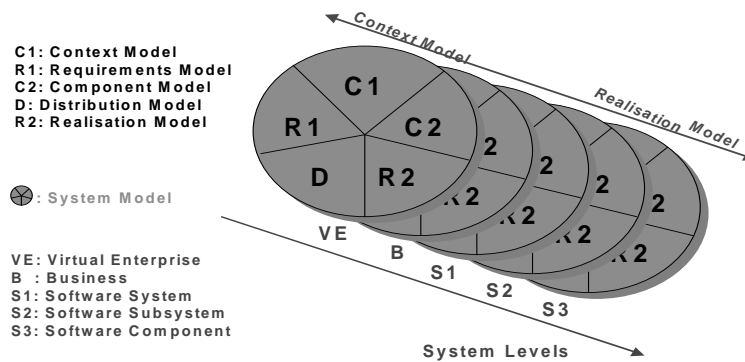


Figure 2-6: System Model Hierarchy - Generic View

The proposed system model can be used to describe a target system at its specific level. As can be seen from the picture, the context model relates the target system to the higher-level system model, while the realization model relates the target system to the lower-level system model. A specific view of this model depicting the C2IS scope addressed in MACCIS II is shown in Figure :

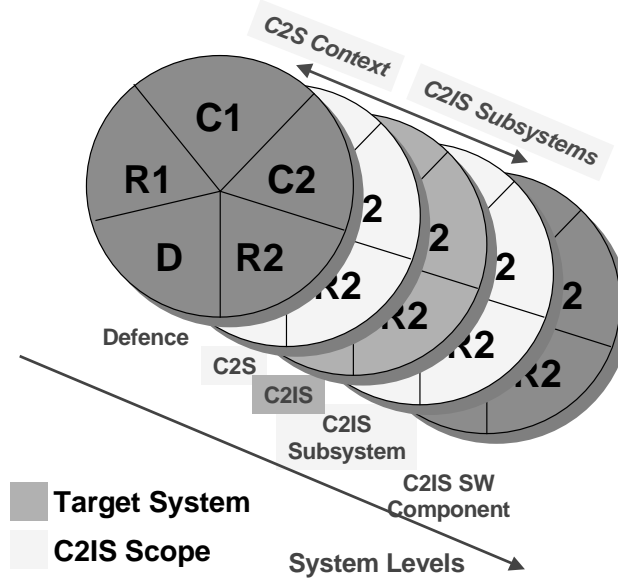


Figure 2-7: System Model Hierarchy - MACCIS II View

2.1.4. Component Technology

The MACCIS II framework is based on the software components and objects paradigm, and the technical architecture thus reflects modern software architectures for software systems, utilizing component and object technologies. Note that the distributed objects paradigm does not mandate that specific distributed objects technology is used; a non-distributed system is just a special case of a distributed system with one node.

The MACCIS II software component architecture is based on the component architecture defined in the COMBINE project [4]. The basis of the component architecture is a generic system architecture, often called reference architecture. The reference architecture defines a set of logical tiers, each of which consists of a set of components. The tiers of the logical architecture give a way of structuring components and systems of components that is very flexible. Each tier contains different kinds of components with certain characteristics. Each tier has specific responsibilities within a distributed system, and so provides one of the necessary separations of distribution concerns.

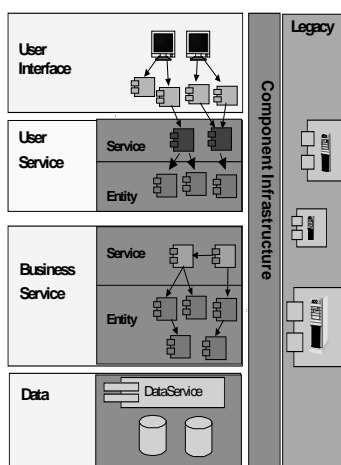


Figure 2-8: Software Component Reference Architecture - MACCIS II View

Figure shows the logical tiers of the software component architecture:

- **User Interface tier:** The user interface tier provides user interaction logic through a set of user interface components. It communicates with the user service tier.
- **User Service tier:** The user service tier provides the user’s model, which may include user session logic and user-side representations of processes and information. It is an abstraction for a set of business services, making the business service provision transparent for the user interface tier.

- **Business Service tier:** The business service tier provides service components that represent business functionality and also pervasive functionality (horizontal vs. vertical services). This tier provides enterprise-level services, and is also responsible for protecting the integrity of enterprise resources at the business logic level.
- **Data tier:** The data tier provides data service components for access and updating data resources. A data service component is the provider of persistent storage services, and provides mechanisms for storing data in some manner.

2.1.5. Modelling Language

The UML language subset that is used for MACCIS II is based on UML 1.4 specification from OMG [5]. The subset is defined by the notational elements used in the essential and subsidiary models and the semantics of those. This includes the following main notational elements.

- Action states, Sub activity states and Activity diagrams
- Actors, Use cases and Use case diagrams
- Sequence diagrams / Collaboration diagrams
- Packages, Subsystems, Classes, Interfaces and Class diagrams
- Components, Nodes and Deployment diagrams

The use of modelling elements and diagrams are further elaborated in the model and process specification.

2.1.6. Baseline Development Process

The baseline development process in MACCIS II is based upon RUP [1 and 2]. The development process is shown in Figure . This view of the process shows the main activities performed during a system development process. The phases of the software development process is specified on the horizontal axis and the set of activities on the vertical axis. The area above the activity lines indicates the effort generally required to specify the different models in each of the phases.

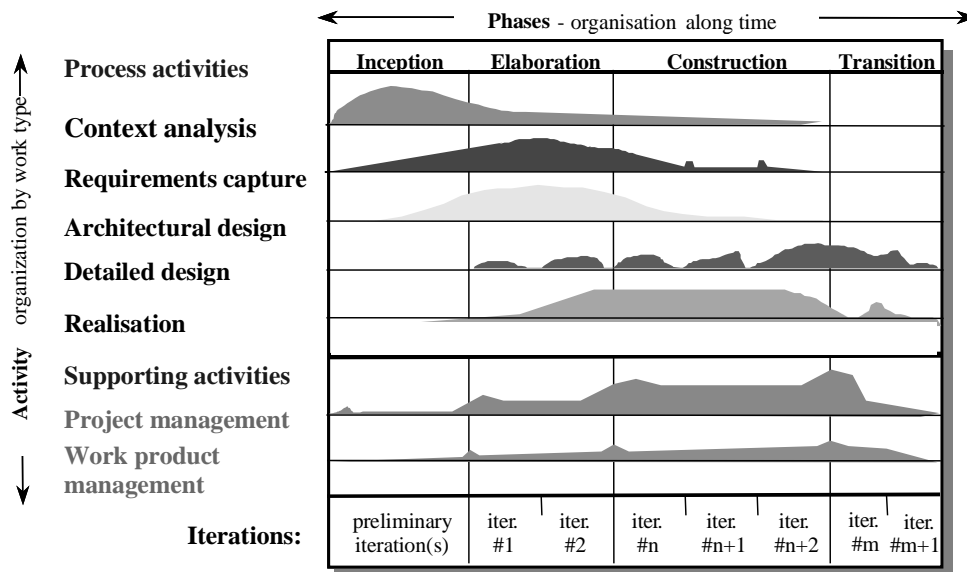


Figure 2-9: Development Process - MACCIS View

There are four phases in the development process: the inception phase, the elaboration phase, the construction phase, and the transition phase. The completion of a phase means that the product under development has reached a certain degree of completeness and thus represents a major milestone of the project.

There are two kinds of activities: Process activities and supporting activities. Process activities are directly related to the system development tasks. Supporting activities support the process activities to ensure that they are carried out effectively.

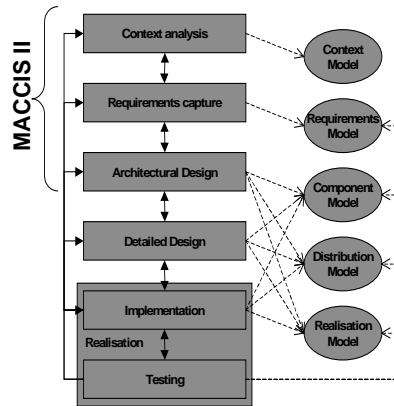


Figure 2-10: Process Activities

Figure shows how each process activity contributes to developing the various architecture models. The MACCIS framework is an architectural framework that primarily addresses the three top activities: context analysis, requirements capture and architectural design. However, the baseline development process also supports some of the more refined activities towards implementation and testing. As can be seen from the figure, the context analysis and requirements capture activities are focused on a specific architecture model, while the architectural design activity contributes to the several architecture models.

2.2. CCIS – system overview

This section outlines a system overview to the Command Control and Information System (CCIS)¹ implemented in the Norwegian Army. Most documentation on CCIS is Restricted pursuant to the Norwegian law of March 20th 1998 number 10 – “*The security law – about preventive security service*”. Hence there will not be any in-depth description of CCIS in this thesis.

The reason for this documentation to be Restricted is that CCIS can carry information classified up to Top Secret.

2.2.1. Functionality and extent

CCIS for the tactical units in the Norwegian Army mainly consist of the following sections:

Operation specific applications

The Operation specific applications will be available for the user. In CCIS the Operation specific applications consist of:

- Command and control
- Fire support
- Logistics support (Ammunition supply)

Support applications

The Support applications are the basic modules in which the Operation specific applications are based on. The Support applications will mostly be based on COTS or if possible reuse services already developed for the Norwegian Defence.

The Support applications in CCIS consist of:

- Map system
- Office support
- Battle log
- Order of Battle (ORBAT)

Information infrastructure

Both the Operation specific- and the Support applications are supported by a set of services in an underlying information infrastructure. Addresses and contracts for information handling are such services. The information is secured and administered with reference to the requirements and procedures laid down by the Norwegian Defence. Tactical Message Handling System (TMSH) is a system already developed for the Norwegian Army and will be used to realize the information infrastructure.

Communication infrastructure

The transfer of information intra modular (within a command post) and inter modular (between command posts) is done by some communication equipment, which is

¹ This documentation is Restricted by Norwegian law and do not have public access. Hence there is no reference to CCIS in the bibliography.

available in every cell and to every user. The Communication infrastructure will basically be made up of existing communication solutions like Tactical Digital Communication (TADCOM), Multi-Role Radio (MRR) and the Norwegian Defence Digital Net (NDDN). Wherever the existing communication infrastructure in CCIS needs to be supplemented, COTS SW and HW will be used in order to reduce the cost. These products must however fulfill the requirements regarding security, environment and availability.

All inter modular data communication is based on TMHS with underlying general IP infrastructure.

2.2.2. Communication infrastructure

TADKOM, MRR and NDDN will be used to link all the units together in a common network. The maneuver battalions are connected to the system via MRR. Hence these battalions will have less transfer capacity than other units physically connected to the system lines.

The command posts (CP) are inter modular connected via different communication networks. This network is based on the existing X.25 standard with an overlying IP level. The general IP network also connects command post to the NDDN.

The distribution of plans and commands will be performed through the stationary and tactical communication facilities. The data transfer is controlled only by the already existing mechanisms in the X.25 and IP network. Hence strict configuration and manual surveillance is needed in order to prevent blocking or errors.

2.2.3. Information infrastructure

The information infrastructure, the message handling and communication are all means to support activities like distribution, changing, storing information and supporting the accomplishment of tasks (work flow).

The information infrastructure is hidden for the regular user. Specially trained technical personnel will configure the system before it is put into operation. During operation the same personnel can reconfigure the system and monitor its status. Operation service and maintenance is strongly related to the status and topology of the communication network.

The same functionality is offered to every command post. But the offered information will depend upon the type of command post and the type of communication. Messages are used to transfer data between the units. The physical position of every unit is automatically updated by means of replication mechanisms. This replication is automatically performed every time a change is made to the database (data push) or every time some information is requested from the database (data pull).

2.2.4. User environment

Every cell in every command post will have the information infrastructure, the Command Post Communication System (CPCS) and TMHS installed. One to five persons attend to every cell.

The users will normally be officers specially trained within the Norwegian Army area of operations. Normally they will not have any special information technology education. The primary use of CCIS is in the field. But CCIS may also be used in an office environment. Within this environment the system will support operative planning and preparations for field drills.

The system will be realized to meet different user demands. Different users will have different requirements regarding response time and reliability, e.g. a fire order requires high reliability and quick delivery.

The system will be used in a mobile environment where the bandwidth will vary. To a limited extent roles and equipment shall be able to be moved around. During small periods of time some parts of the network may be isolated. Hence these parts need to be updated (synchronized) when they are restored. The availability of services and resources is therefore a very important aspect.

2.3. DOLCCIS

The domain language for command and control information systems (DOLCCIS) [6] is the vocabulary that domain experts (typically users, field operatives) and developers should be able to communicate ideas, wishes and needs that are difficult to express using a normal spoken language. It includes well-defined domain concepts, notation and a set of models. The set of models aims to describe the domain in different views, which together constitute a complete domain model. The domain language is used to perform domain analysis. In addition to being a communication channel between users and developers, it is a formalization of how to describe system requirements for operations.

DOLCCIS is a refinement of MACCIS enterprise viewpoint. It is a generic language using a subset of UML to support modelling of C2IS, i.e. it provides the general mechanisms to model the enterprise viewpoint and specific mechanisms that are helpful in the C2IS domain.

The domain modelling language will consist of a set of models and means for achieving those models. The modelling language can be compared to a natural language, e.g. Norwegian, where letters are used to compose words, and structure sentences by certain rules to accomplish meaningful semantics. Similarly, in the domain language, modelling elements (e.g. classes, relations) are used to compose models, and structure them to accomplish meaningful semantics. In addition to standardizing the generic language, framework for supporting user-oriented, high-level, and domain-focused modelling concepts will leverage the ability of the user/operatives to capture and communicate ideas to domain analysts and system developers. The primary concepts considered are domain-oriented object models and standardized symbols for the army.

The objectives are to:

- Specify a domain language for the army,
- Define a minimal set of models from MACCIS enterprise viewpoint (and possible extensions) that are needed for domain analysis,
- Describe how these models should be used, and
- Identify the basis for a framework for identification and application of domain concepts and symbol representations that are commonly in use in the army domain.

It is outside the scope of this report to describe process and method related the process of domain modelling. Process and method is described by the activities on system and model development process. DOLCCIS pursues a generic approach for domain modelling, focusing on supporting command and control aspects of the army. It is not limited to the operative aspects of the army and may also be applied within the areas of strength production and material management.

2.4. Definition of concepts

This section describes some concepts that are used throughout this thesis. These are well-known concepts like Rational Unified Process (RUP) and Reference Model of Open Distributed Processing (RM-ODP).

2.4.1. Network centric defence

Up to now the Norwegian army have been a so-called platform centric defence where, e.g. army units, frigates and combat aircrafts have been in focus. The goal is to leave this kind of defence and move towards a network centric defence and finally reach an information centric defence.

A network centric defence has its focus on the decision makers, instruments and sensors in a way that connects these three aspects in order to provide an optimal situation and resource information.

The information centric defence uses the information as the primary instrument during information operations. The information will be crucial in order to make the right and cost effective decisions along with optimal exploit of resources.

Figure 2-11 below shows the three different defences. The platform centric to the left, the network centric in the middle and the information centric to the right.

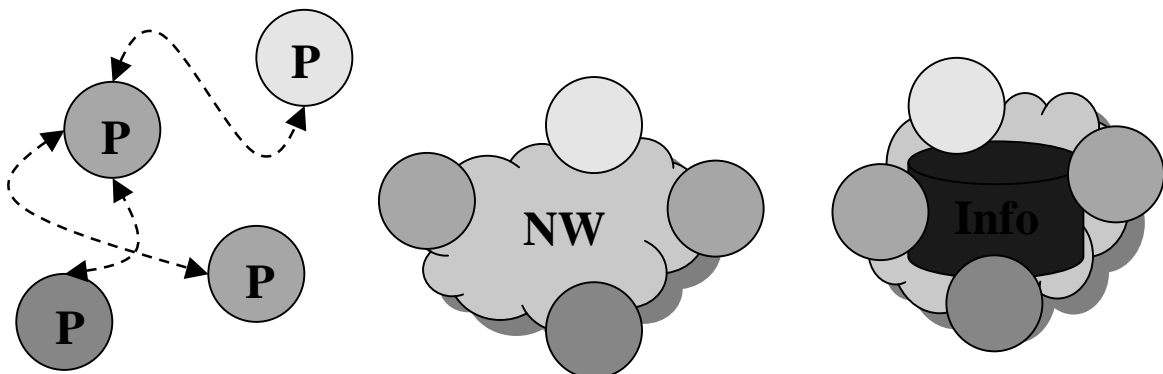


Figure 2-11: Platform-, network- and information centric defence

This development perspective is not an absolute condition for this thesis, but it outlines a probability for what will happen in the coming years. This readjustment of the Norwegian army will go far beyond the ongoing change, which focus on personnel reduction and detailed discussions about different platforms.

2.4.2. Rational Unified Process

Rational Unified Process (RUP) [7] is a *software engineering process*, aimed at guiding software development organizations in their endeavors. The RUP has a very well defined and regular structure, using an object-oriented approach for its description. The RUP is also a *process framework* that allows a software development organization to tailor or extend the RUP to match its specific needs. It captures many

of modern software development's best practices harvested by Rational over the years, in a form suitable for a wide range of projects and organizations.

Figure shows the overall architecture of the Rational Unified Process. RUP has two structures, also referred to as dimensions:

- The horizontal dimension represents time and shows the lifecycle aspects of the process as it unfolds.
- The vertical dimension represents core process disciplines (or workflows), which logically group software engineering activities by their nature.

The first (horizontal) dimension represents the *dynamic aspect* of the process expressed in terms of cycles, phases, iterations, and milestones. In the RUP, a software product is designed and built in a succession of incremental iterations. This allows testing and validation of design ideas, as well as risk mitigation, to occur earlier in the lifecycle. The second (vertical) dimension represents the *static aspect* of the process described in terms of process components: activities, disciplines, artifacts, and roles.

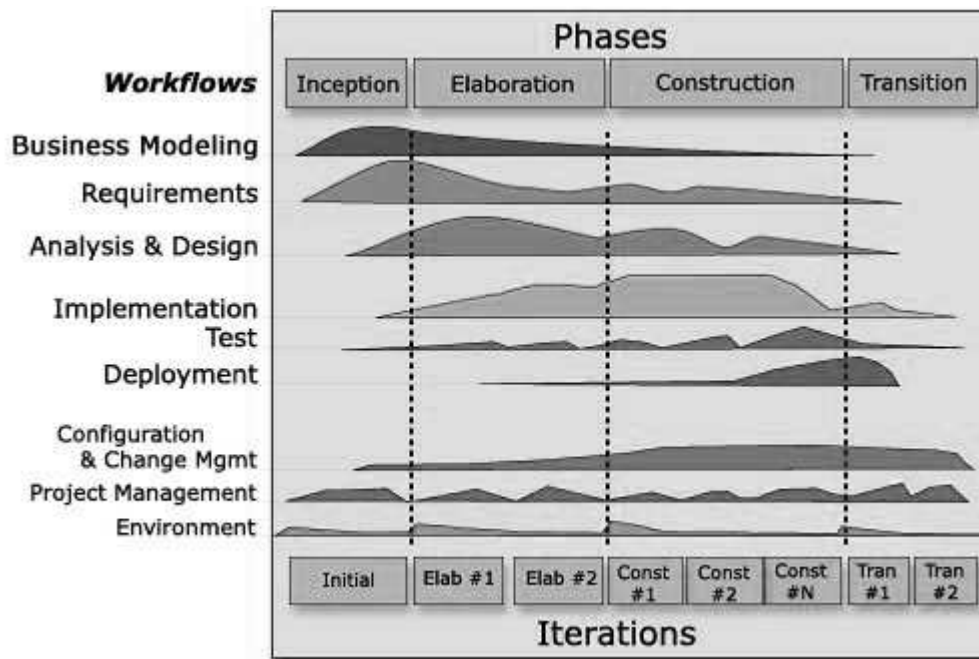


Figure 2-12: RUP development phases

As mentioned above the Rational Unified Process is also a *process framework* that can be adapted and extended to suit the needs of an adopting organization. It is general and comprehensive enough to be used "as is" by many small-to-medium software development organizations, especially those that do not have a very strong process culture. But the adopting organization can also modify, adjust, and expand the Rational Unified Process to accommodate the specific needs, characteristics, constraints, and history of its organization, culture, and domain. A process should not be followed blindly, generating useless work and producing artefacts that are of little added value. Instead, the process must be made as lean as possible while still fulfilling

its mission to help developers rapidly produce predictably high-quality software. The best practices of the adopting organization, along with its specific rules and procedures, should complement the process.

2.4.3. Reference Model for Open Distributed Processing

Reference Model for Open Distributed Processing (RM-ODP) [3] is a joint standardization activity by both ISO and ITU-T. RM-ODP adopts an object-oriented approach for the specification of distributed systems.

RM-ODP uses something called viewpoints to partition a system specification into a number of different components. Each component (i.e. viewpoint) is a complete and self-contained description of the required distributed system targeted towards a particular audience. The terminology (language) used for this description is therefore tailored towards this target audience.

Example: Viewpoints analogy

To illustrate the concept of viewpoints further, consider a non-computing example, e.g. different viewpoints on a civilian aircraft. One viewpoint is that of the manager responsible for the operation of the aircraft. This manager views the aircraft as an artifact with a given lifetime, requiring maintenance at certain intervals at a certain cost and capable of carrying passengers on certain routes at a given profit margin. The maintenance engineer, on the other hand, views the aircraft as a set of interconnected components with given technical specifications and testing regimes. Finally, an air traffic controller will view an aircraft as an object occupying a particular location in the airspace at a given time and flying on a pre-determined route. Each participant will have a terminology for discussing the aircraft to colleagues in an unambiguous manner.

Viewpoints are as central to RM-ODP as the seven-layer model is to the ISO standard for Open Systems Interconnection (OSI). It should be stressed however that viewpoints are not layers. Rather, they are projections on to the underlying system. This distinction is depicted in Figure 2-13 and Figure 2-14 below.

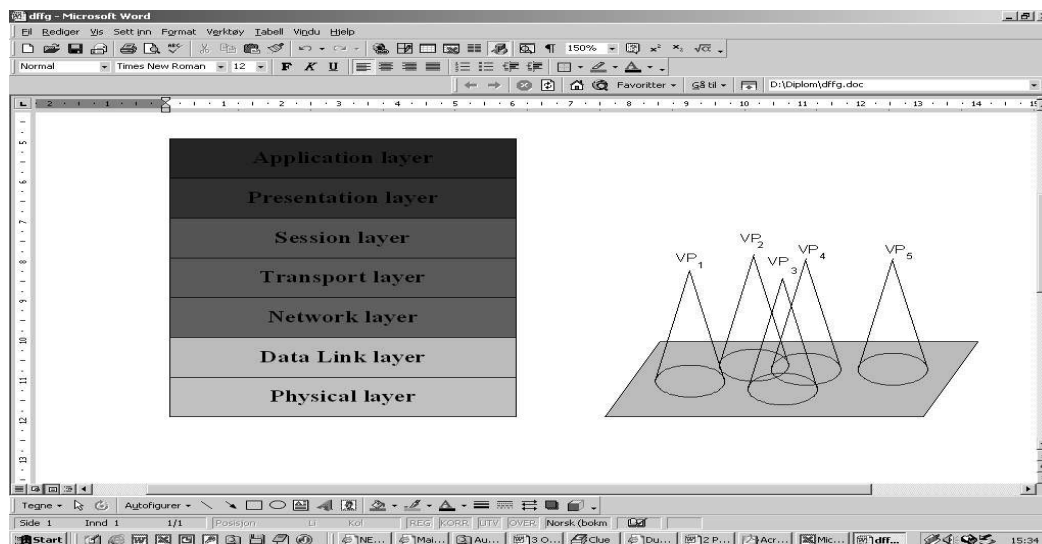


Figure 2-13: Layering and OSI

Figure 2-14: Viewpoints and RM-ODP

RM-ODP defines five viewpoints, namely the Enterprise, Information, Computational, Engineering and Technology Viewpoints. Each viewpoint has a corresponding viewpoint model.

More detail on the five viewpoints and associated models is given below:

i. Enterprise Viewpoint.

The Enterprise Viewpoint considers the role of the distributed system in the operation of an enterprise. The aim of the corresponding specification is to model the scope and objectives of the system. The main concept behind the Enterprise Viewpoint is that of a contract expressing the obligations of the various participants in the distributed enterprise. The Enterprise Model then supports the specification of such contracts. Key concepts in the Enterprise Model include *agents, artifacts, roles, communities* and *federation*.

ii. Information Viewpoint

The Information Viewpoint considers a distributed system from the perspective of the information content. More specifically, it considers the various information elements in an organization, the flow of this information around the organization and processes for manipulating this information. The corresponding model defines the basic concepts of objects and composite objects; it then enables the specification of information schemata over these objects. Three styles of schemata are defined: *invariant schemata* which describe relationships which must always be true in the specification, *static schemata* which describe assertions which must be true at a single point in time, and *dynamic schemata* which define how the system should evolve in terms of changes to the environment and to individual objects.

iii. Computational Viewpoint

The Computational Viewpoint considers the logical partitioning of the distributed system into a series of interacting entities. This partitioning is logical in that it does not imply any particular realization in a distributed environment, e.g. mapping to nodes or address spaces. The corresponding model adopts an object-oriented approach (a specialization of the model presented earlier). An application is decomposed into a number of interacting objects with each object offering one or more interfaces. All interaction is via these interfaces. In order to interact, it is necessary to have a binding to the target object(s). Three styles of binding are supported: *operational binding* (supporting the invocation of operations), *stream binding* (supporting continuous media interactions), and *signal binding* (supporting real-time processing). The application developer can also specify a set of constraints on the underlying implementation of the object through an environmental contract. This contract includes specification of the required level of distribution transparency and the quality of service offered by the interface. Such specifications are declarative in that they say what is required rather than how it is to be achieved. The latter issue is the concern of the Engineering Viewpoint as discussed below.

iv. Engineering Viewpoint

The Engineering Viewpoint considers the infrastructure required to support the physical realisation of a distributed application in terms of both communications and end systems technologies. The Engineering Model defines a basic set of abstract concepts required to model communications and end system resources. In terms of communications, the model defines the concept of a channel, which consists of a configuration of stubs, binders and protocol objects. In terms of end system resources, the model defines the concepts of nodes (as physical machines), capsules (as an execution environment for objects) and clusters (as a migratable set of objects). In addition, the model defines transparency functions as a means of achieving a given level of distribution transparency; there is a direct translation between the requirements specified in the environmental contract (see above) and the set of transparency functions provided.

v. Technology Viewpoint

The Technology Viewpoint considers the development of a distributed system from the perspective of the identification, procurement and installation of particular hardware or software technologies. This viewpoint effectively grounds the overall design in terms of realizable technologies. The corresponding model is used to justify rather than to describe the selection of particular technology artifacts. In particular, the Technology Model enables the technology provider to provide a declaration that a particular technology will meet its requirements in terms of Implementation eXtra Information for Testing (or IXIT).

The different viewpoints outlined above effectively create a separation of concerns in the specification of a system. The five viewpoints collectively provide a complete specification of the system. Note however that some issues will appear in a number of different viewpoints. For example, consider the provision of security in a system. In the Enterprise Viewpoint, the specification might state that a particular project should

operate with a "need to know" policy, i.e. each project manager should know the minimal amount of information for the goals of the project to be met. In the Information Viewpoint, this would translate into constraints on the schema offered to different participants. At the Technology Viewpoint, a decision could be taken to adopt a specific security service to implement the desired policy.

3 DISCUSSIONS

3.1. CONNECTIVITY	28
3.1.1. INTEROPERABILITY	28
3.1.2. TRACEABILITY	29
3.2. COST EFFECTIVITY.....	30
3.2.1. NUMBER OF MODELS	30
3.2.2. APPLICATION OF MODELS ON REAL-WORLD PROBLEMS	31
3.3. SYSTEM DEVELOPMENT	32
3.3.1. DEVELOPMENT TOOL.....	32
3.4. SECURITY	35
3.5. PARALLEL DEVELOPMENT.....	36

3. Discussions

These sections present findings and discussions around these findings according to the three initial objectives for this project.

The first objective has been to look at connectivity with emphasis on traceability, i.e. how well are the different models in the different views connected to handle traceability.

The second objective has been to look at cost effectivity, i.e. whether or not the framework is able to produce what it is supposed to do and whether or not the framework is not producing things it is not supposed to do.

The third objective has been to look at system development with emphasis on whether or not the framework is following any special development process.

3.1. Connectivity

The next two sections will look at connection in terms of interoperability and traceability. Section 3.1.1 looks at how MACCIS II is related to NATO approved standards, and section 3.1.2 looks at traceability among models, among different versions of the same model and among corresponding models in different sub-activities.

3.1.1. Interoperability

In MACCIS II it is emphasized that there are two goals in the development of the model set. One is to describe an existing system in order to achieve interoperability with other systems. The second is to describe a system under development to assure the system is built correct. According to [8] one huge challenge is that NATO and USA do not use the same models as specified in MACCIS II.

In order to achieve these two goals this architectural framework has to be extended and “sold” in two steps.

First it has to be sold to the remainder of The Norwegian Defence. There is no point in “selling” outside The Norwegian Army unless the framework is adapted in the entire organization. Every single member in the organization has to be familiar with notation and structure in order to apply the framework onto real-world problems.

Second the framework has to be “sold” to NATO. At this level there are two main competitors, *NATO Command, Control, Communications Technical Architecture* (NATO C3 TA) and *Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance Architecture Framework* (C4ISR AF). NATO has approved both.

Today MACCIS II only includes CCIS. But in order to compete with the frameworks mentioned above MACCIS II has to be extended to include C2S, i.e. include

doctrines, personnel, methods, procedures, equipment and facilities. The focus should maybe be on goals and objectives for the business the IS is supposed to be applied in instead of focusing on the IS alone.

Again, according to [8], MACCIS II has an advantage in its support of the futuristic vision about a flexible, dynamical and interoperable network centric defence. This advantage is reflected in the component models. These models attend to the connection between the enterprise aspects, i.e. vision, purpose, goal and tasks, and the functions and components that are supposed to support these aspects. These connection-describing models will be essential in the development of the futuristic network centric defence. See chapter 2 for more details on different defence levels.

3.1.2. Traceability

In order to get good consistency between the different models it is necessary to have traceability mechanism among the developed models, among different versions of the same models and between corresponding models in other sub-activities.

Traceability among the developed models

i.e. different requirements applied on one model should be reflected in every other model this is related to.

MACCIS II has a description that says in which order the different models are to be developed. But what is missing is a description of how to maintain the traceability among the models.

Rational Unified Process has come up with an explicit traceability model, which indicates the relations between Use Cases and Use Case realizations. However, this is not a relation that propagates as models are being refined. This may lead to difficulties in seeing which superior model elements where the basis for later refinement.

Traceability between different versions

Another, but related problem, is traceability among different versions of one model. Since MACCIS II is following a baseline development process based on RUP, which is iterative, it is a fear possibility that already developed models will be altered because of faults, lack of details or a combination of these. This alteration leads to new versions of the set of models and hence the needs for traceability among different versions arise.

Traceability between corresponding models in other sub-activities

With a uniform application of each model on the different sub-activities, it could be possible to detect similarities in independent sub-activities on every level. As mentioned in section 3.2.2 this detection leads to a more efficient and a more cost reduced development.

Either these problems need either to be solved within the scope of MACCIS II or MACCIS II needs to be extended to involve descriptions of these problems. And since MACCIS II does not have a complementary description of these issues, the latter proposal will be the best solution. In addition this issues a requirement to the development tool, e.g. Rational Rose, in order to deal with these problems. See section for 3.3.1 more details on development tool.

3.2. Cost effectivity

If framework architecture is to be cost effective it has to meet all the requirements initially made for the framework to solve, but no more than that. In order to keep focus and not implement all sorts of outside interference, the latter “... no more than that” is important. It is up to the framework not to propose models and other elements that may get things out of focus.

If this framework is supposed to be applied on all levels in the Norwegian Army it is extra important that the framework is hands-on, direct and easy to understand. This is because the number of operative trained personnel outnumbers the technical educated personnel and hence most people do not know the technical terminology often used in such framework.

3.2.1. Number of models

MACCIS II presents different models in five different views. If only the essential models are taken into account there are about an average of four models on each view and a whole lot of Use-cases to refine into Interface models, Security models, Distribution models and so forth. This gives us an informal rule:

M_a – average number of models in each view

V_t – total number of views

U_t – total number of Use-case

O – a undetermined number of descriptions and supporting models

Rule: $(M_a * V_t * U_t) + O$

For a small system with only five Use-cases this set has scaled to about 100 models, which already is a great amount. It will take a lot of time and effort even for a trained technical person to get a good overview in this jungle of models. In addition there has to be a description of every model, a traceability description/model of the requirements and maybe some supporting models to clarify some situations. This gives a vast number of models for only a small part of maybe the entire enterprise.

If this framework is applied on every level in the Norwegian Army, the operational trained personnel will only be affected by the Enterprise viewpoint. But even here, with the situation described above, the number of models will scale up to about 30 models and then some overhead.

In this graduate thesis this is seen as a weakness because of the vast size and danger for non-technical personnel to loose track and overview. The number of models proposed in every view should be revised and preferably reduced. This revision of the models is however outside the scope of this thesis, but it is recommended in chapter 4 as further work.

3.2.2. Application of models on real-world problems

If a framework is to be used in the same way by every one, it has to contain practical examples of models with comments that enhance the essence. Practical examples and belonging comments will force an approximate uniform application of the models on real-world activities. This will further lead to cost and time reduction since only one module need to be implemented and only reused in other similar activities. Without this practical approach the uniform modelling is lost. Hence the traceability between two or more activities at any level is lost, as described in section 3.1.2.

3.3. System development

The development process proposed in MACCIS II is based on the baseline development process in RUP. See section 2.1.6 and 2.4.2 for further details. For MACCIS II to support modern software development, which RUP does, there are some important *best practices* [7] it has to fulfill:

- Develop software iteratively.
There are a number of reasons to adopt an iterative approach. Some of them are the ability to taking into account change in requirements, the possibility to mitigate risks as they are discovered little by little, the possibility for tactical releases and the flexibility to change the development process along the way.
- Manage requirements.
Some benefits to effective requirements management are better control, improved software quality, cost- and time effective and improved team communication and understanding.
- Use component-based architectures.
Some benefits in component-based architecture is component reuse and use of off-the-shelf components for various domains.
- Visually model software.
The advantage here is in giving a conceptual understanding of the system, and hence getting an overview in order to comprehend the complex system.
- Continuously verify software quality.
The iterative approach offers both product and process quality because both qualities may be improved in the next iteration.
- Control changes to software.
This includes requirement management, but also management of changes in design and implementation.

All these best practices can be summarized in iterative development and good management. As these *best practices* outline, an important aspect is the ability to change everything from the requirements to the development process in the next iteration. This requires a very good mechanism for traceability as described in section 3.1.2.

3.3.1. Development tool

Based on experience Norwegian Defence Research Establishment [8] have come up with some significant factors that must be used in the evaluation of a development tool to support modelling in MACCIS II:

- Should be easy to use! This is important if non-technical personnel, i.e. operative personnel, shall be able to use the tool without heavy education and practice.
- Support for object-oriented and component-based development. This is obvious since MACCIS II is based on an object-oriented and component-based development process.

- It should be able to represent all the models specified in MACCIS II. A tool that is not able to represent the models required by a framework architecture is worthless.
- It has to support traceability among models. This is discussed in section 3.1.2.
- It has to have multi-user function. This is to support cooperative work between everyone involved.
- Support for detail refinement, i.e. hide some details at a higher level and describe them at a lower level. With this approach it is much easier to get an overview and to differentiate superior overview and more details needed only by domain experts.
- Relations (traceability) among different models should show in a superior architecture.
- The tool has to provide some sort of version-control, e.g. in order to look back and see why a change was made.
- The tool should have a repository to represent the models.
- The tool should be compatible with other tools. E.g. a model will be used in a Microsoft Power Point representation or in a Microsoft Word Document.
- Adaptability to MACCIS II, i.e. an ability to create templates to adapt to the models.
- The same tool should be able to support the development of both the enterprise architecture and the system architecture. If not there may be severe problems in converting from one architecture to another. It may lead to inconsistency, loss of details etc.

Further [8] summarize some experiences made on Rational Rose:

1. Rational Rose has good support for analysis, design and implementation of object-oriented and component-based systems.
2. By experience it seems that most of the models can be represented. But the following problems are found:
 - a. En-3: Trouble in representing components in physical nodes in the "Deployment model" in the Engineering viewpoint.
 - b. EV-3, EV-4, EV-12: It takes a lot of resources to develop all the military signs required for the enterprise modelling. Rational Rose has the possibility to define new "stereotypes", but the process at doing so is far too comprehensive.
 - c. EV-12: It is impossible to represent the same role, e.g. G-3, in different packages. Only classes that are physically present in a package can be viewed as "compartment" of a package.
3. RUP is recommending "Traceability" diagrams in order to show the relation between "Use Cases" and "Use Case realisations". In relation to MACCIS II a "Use Case realisation" has to be developed in order to show the traceability between "Enterprise viewpoint" and "Information viewpoint"/"Computational viewpoint". The reason to develop this realisation is to outline how a requirement, in the form of a "Use Case", is realized in the "Information viewpoint" and "Computational viewpoint".

All the diagrams related to a “Use Case” realisation are then placed under its respective “Use Case realisation”. Further a “realisation relationship” is made between the “Use Cases” and the belonging “Use Case realisations”.

4. There are several supporting possibilities in the development of models in parallel:
 - a. The model can be decomposed into units called ”controlled units”.
 - b. Rose allow model files and ”controlled units” to be moved between workplaces by the means of a virtual catalogue map.
 - c. Rose allow integration with standard version control systems like Microsoft Visual SourceSafe
 - d. A tool named ”Model Integrator” can compare and merge models with ”controlled units”.
 - e. Rose can be integrated with Microsoft Repository. This allows other modelling tools to import any model published from Rose to the repository.
5. It is possible to decompose models by the means of packages and sub-diagrams. However, relations do not propagate either up-wards or down-wards in these structures. On the other hand, relations between classes described in one diagram will automatically be reproduced in every other diagram where these classes are used.
6. Many other tools support import of some Rational Rose models. However, experience shows that some parts of the models seem to vanish during such imports.
7. The flexibility to create ”templates” in order to adept to the models in MACCIS II is rather low in Rose. Every time a new project is made Rose establish a permanent structure of packages: ”Use Case view”, ”Logical view”, ”Component view” and ”Deployment view”. Any enhancement in this structure will mostly be placed under ”Use Case view” and/or ”Logical view”.
8. Rational Rose is not easy to use. This implies a great focus on education and support from the supplier if this tool is to be introduced to the entire military force. There are also some illogical ways to configure the models under development. This often makes the models untidy and gives them low readability.

3.4. Security

As mentioned in chapter 2, MACCIS II is based upon RM-ODP. RM-ODP [3] defines a number of functions, which are necessary to support a comprehensive RM-ODP platform. Functions fulfill a number of different purposes in RM-ODP. For example a set of functions support the implementation of security (including the functions of access control, authentication, key management and security auditing). Table is listing all the security functions provided by RM-ODP.

Security	Access control	Prevents unauthorized interaction on interfaces
	Security audit	Monitors and collects security information
	Authentication	Confirms identity of an object
	Integrity	Prevents unauthorized creation, alteration or deletion of data
	Confidentiality	Prevents unauthorized disclosure of information
	Non-repudiation	Prevents the denial of an object from having participated in an interaction
	Key management	Manages cryptographic keys

Table 3-1 Security functions in RM-ODP

MACCIS II has a model in the Engineering viewpoint (see Figure 2-3) called *System Security Model (En-1)*. The purpose of this model is to describe the different security mechanisms that are used in the system. This model depicts security policies, security requirements and information classification. These constraints are defined in models in Enterprise viewpoint, Information viewpoint and Computational viewpoint. However, MACCIS II do not describe how these constraints are extracted from the three latter mentioned viewpoints. That constitutes a great danger in not intercepting the most essential security menaces. And as mentioned in section 2.2, CCIS is classified up to Top Secret information. Therefore security must be of great importance and any security menace has to be found and solved.

The System Security Model comprises the functions provided by RM-ODP and a set of UML diagrams describing how these security mechanisms are used in the system.

3.5. *Parallel development*

As can be seen in chapter 2, both CCIS and the first version of MACCIS were developed in parallel with start-up in 1998. The ideal situation would be to have a complete and finished version of MACCIS before the industry began its work on CCIS. In that case the Norwegian Army could by a contract agreement demand the industry to document CCIS in conformance to MACCIS. But with this real situation, the industry already had begun to document CCIS in their own way. Hence the first deliverable of CCIS did not get documented in conformance to MACCIS.

This situation has put some limitations on this project. In connection with CCIS there exist no physical documentation made by the industry, which is in conformance to MACCIS. Hence there exist no practical application of MACCIS and therefore no experience data.

In posterity MACCIS has been applied on small projects and used in the modelling of small parts of CCIS. The Norwegian Defence Research Establishment (NDRE) [8] has made a report based on experiences obtained using MACCIS for modelling CCIS. Note that also this report is based on the evaluation of the first version of MACCIS, not MACCIS II.

4 CONCLUSION AND FURTHER WORK

4.1.	CONCLUSION.....	38
4.2.	FURTHER WORK	39
4.3.	PERSONAL EXPERIENCE AND OPINION.....	40

4. Conclusion and further work

This chapter includes a conclusion in section 4.1, a discussion on further work in section 4.2 and some thoughts about personal experience acquired during the project in section 4.3.

4.1. Conclusion

The initial objectives for this project have been threefold. The first objective has been dealing with connectivity, in terms of interoperability and traceability. Interoperability is concerned with how well MACCIS II is interoperable with other architectures both within The Norwegian Defence and in NATO. The traceability has been seen as traceability among developed models, traceability between different versions of the same model and traceability among corresponding models in other sub-activities.

The second objective has been concerned with cost effectivity, i.e. whether or not the architectural framework is fulfilling its initial goals. Two aspects have been discussed. One is the number of models and whether or not that number should be reduced. The second aspect was concerned with how the different models should be used on real-world problems.

The third, and last objective, has been concerned with the development process used in MACCIS II and some significant factors that should be used in the evaluation of a development tool meant used for MACCIS II.

All these three objectives have been discussed with great support in [MACCIS P&F]. But even that report is not yet completed. Hence the background experience on real-world problems is insufficient. Other parts of the discussions are based on undocumented experience and informal talks with people trying to apply MACCIS II on actual problems. Many of these expressions of opinion are in conformance to the more formal experience report developed by Norwegian Defence Research Establishment [8].

The idea and purpose of an architecture framework just like MACCIS II is indeed great. It proposes a uniform way of representing an architecture, which offers advantages like a well-known representation and understanding and different interfaces described with the same models. But there are many pitfalls on the way during the development of a framework. MACCIS II is in lack of good examples included comments on how constraints from one or more models shall be extracted and put into another model, e.g. the problems related to the *System Security Model*. It is also in lack of good mechanisms for traceability. And it should aim at a reduced number of models because of the total complexity. The problem concerning development tool is also important and remains solved and decided upon at a higher level in the organization. MACCIS II is said to have a slight problem in its enterprise modelling. It does not take into account the representation of actors with belonging roles moving around in the organization. This together with the development tool discussion brings out a big practical problem. Today this problem is solved in quite various ways, depending on the tool and the person doing the modelling.

By highlighting these objectives I hope the discussion and improvement of MACCIS II continues. As mentioned above, the idea and most of MACCIS II is very good. But all the practical and consistent problems needs to be solved before the framework is “sold” to the entire Norwegian Army. The problems need to be sold to prevent local adaptation.

4.2. Further work

This section lists some aspects, which need further attention in order to make MACCIS II ready for the huge organization the Norwegian Army is.

- Revision of models (see Figure 2-3) is necessary to see weather or not they all are useful and needed. To do so a small group of domain experts (operational trained personnel) and modelling experts have to complete the evaluation report [P&F]. This report has basically done evaluation of *Enterprise viewpoint* and *Information viewpoint*. Evaluation of *Computational view*, *Engineering view* and *Technology view* remains. Hopefully this revision leads to a reduction in number of models.
- The discussion on development tool needs to continue. There do not need to be only one tool but every tool should fulfill all the significant factors required by MACCIS II. This to prevent local adaptation, which may lead to large variations in the representation of models.
- The security aspect should be revised. It is very hard to say if MACCIS II is able to catch all the security hazards present in a system like CCIS and in the Army as a whole. One thin is clear, it does not explain how security constraints are extracted from one model and put into another at a different viewpoint.
- The above aspect about security also has to do with the traceability among models. In general MACCIS II needs a better traceability model integrated in the model structure.
- The above aspect about traceability outlines another problem, of which MACCIS II do not have sufficient examples of real-world problems and sufficient explaining comments on each model.
- The strategy of how to implement MACCIS II and sufficient development tools into the Norwegian Army has to be planed. What knowledge level do people working with MACCIS II need to have and how much education do they need to get there. This is a process with a lot of resources like money, time and personnel involved.

4.3. *Personal experience and opinion*

First of all this project has been a valuable experience that I can take advantage of when the fall comes and the employee existence appears. The skill of working independent of others has improved. But also the ability to seek people with the right knowledge and communicate with them has been an experience of great value.

To be a part of an ongoing discussion and work, such as the architecture program in the Norwegian Defence, has been very interesting. The invitation to a workshop in connection with this architecture program and all the talking to different people with different opinions have been both interesting and inspiring. However, this really indicates how fragile this subject is.

5 GLOSSARY

5.1.	MILITARY SPECIFIC ACRONYMS.....	42
5.2.	GENERAL ACRONYMS.....	43

5. Glossary

Section 5.1 explains the full meaning of the military acronyms used throughout this project. Section 5.2 explains the full meaning of the general acronyms, i.e. acronyms that has nothing to do with the military domain.

5.1. *Military specific acronyms*

C

C4ISR AF	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance Architecture Framework
CCIS	Command, Control and Information System
CP	Command Post
CPCS	Command Post Communication System

D

DOLCCIS	Domain Language for Command, Control and Information System
---------	---

F

FLO/LAND	Forsvarets Logistikk Organisasjon avdeling land
----------	---

H

HFK	Hærens ForsyningsKommando (Now FLO/LAND)
-----	--

M

MACCIS	Minimal Architecture for Command, Control and Information System (also called Model-based Architecture for CCIS)
MRR	Multi Role Radio

N

NATO	North Atlantic Treaty Organisation
NATO C3 TA	North Atlantic Treaty Organisation Command, Control and Communications Technical Architecture
NDDN	Norwegian Defence Digital Network

T

TADCOM	Tactical Digital Communication
TMSH	Tactical Message Handling System

5.2. *General acronyms*

I

IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector

O

OMG	Object Management Group
OSI	Open System Interconnection

R

RM-ODP	Reference Model for Open Distributed Processing
RUP	Rational Unified Process

S

SINTEF	The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology
--------	---

U

UML	Unified Modelling Language
-----	----------------------------

BIBLIOGRAPHY

- [2] Elvesæter, B., Aagedal, J. Ø., Berre, A. J., Neple, T., *MACCIS part A – Overview and guide to use the MACCIS framework Version 2.0*, Oslo, SINTEF Telecom and Informatics, 15 December 2001
- [3] Aagedal, J. Ø., Elvesæter, B., Berre, A. J., Neple, T., Oldevik, J., Solberg, A., *MACCIS part B – Specification of the MACCIS framework Version 2.0*, Oslo, SINTEF Telecom and Informatics, 15 December 2001
- [4] Blair, G.S., Stefani, J.B., *Open Distributed Processing and Multimedia*, Boston, Addison-Wesley, 1997
- [5] The Open Group, Western Geco, Open-IT, Softeam, IONA Development, SINTEF, INSEC, <http://www.opengroup.org/combine/>, 2000-2002
- [6] OMG, *Unified Modelling Language (UML) 1.4. Specification*, formal/01-09-67, 2001
- [7] Oldevik, J., Solberg, A., Neple, T., Aagedal, J. Ø., *DOLCCIS – Domain language for command and control information systems Version 2.0*, Oslo, SINTEF Telecom and Informatics, 04 September 2001
- [8] Kruchten, P., http://www.therationaledge.com/content/jan_01/f_rup_pk.html, 2001
- [9] Bednar, I. B., *Evaluation of experiences obtained using MACCIS for modeling C2I systems*, Kjeller, Norwegian Defence Research Establishment, 21 January 2002