Analysis and specification of a Maintenance Information Communication System (MICS), using a modeling approach
remote monitoring in the Dutch power system

Bakker, N.J.W.

Award date:
2000

TU/e technische universiteit eindhoven

Faculty of Electrical Engineering
Section of Information and Communication Systems (ICS/EB) 741

Master's Thesis

Analysis and Specification of a Maintenance Information Communication System (MICS), using a Modeling Approach

Remote Monitoring in the Dutch Power System

By N.J.W. Bakker

Author: N.J.W. Bakker (ID: 392250)

Institute: Eindhoven University of Technology (EUT)
Department of Electrical Engineering
Information and Communication Systems Group (ICS)

Advisors: dr.ir J.M. Wetzer (KEMA)
dr.ing. P.H.A. van der Putten (EUT)

Supervisor: prof.ir. M.P.J. Stevens

Date: June 2000

Summary

Specification and modeling of a Maintenance Information Communication System (MICS) are the central points in this report.

Energy companies need to reduce their integral costs, especially since the 1st of January 1999, when the Dutch government liberalized the electricity market. Integral costs include costs of maintenance, repair and depreciation.
One of the ways to accomplish this reduction is by improving the maintenance system. Both 'Condition Based Maintenance' (CBM) and 'Reliability Centered Maintenance' (RCM) are examples of 'Just-In-Time Maintenance' (JIT-M). On the one hand, JIT-M can contribute to reducing maintenance costs, while on the other hand, it adds greatly to expanding the life-span of equipment, saving costly repairs. Depreciation costs will then decrease, as a result of the possibility to postpone replacements and new investments.

The detailed information on the equipment's condition, that is necessary to realize either CBM (Condition Based Maintenance) or RCM (Reliability Centered Maintenance), may be obtained by 'Remote Monitoring' (RM). As a result, a large number of existing substation- or system-level services and resources can be shared. Efficiency makes the step towards a centrally managed maintenance system more attractive. In addition, the system's possibilities can be extended by introducing new system-level services. Although many benefits of RM are addressed, aspects like security and real-time behavior should be paid attention to.

System requirements have been captured. The resulting raw requirements are fitted into a 'System Requirements Specification' (SyRS), based on an IEEE standard. The SyRS is an aid in finding the system's capabilities and restrictions, although the limited time has not allowed drawing up a complete specification document. During design, eventually, decisions are made on the basis of the SyRS.

Next to specification, 'system modeling' has been an important issue in this report. An object oriented modeling methodology, 'Software/Hardware Engineering' (SHE), has been used to specify the system's behavior, using a dedicated 'Parallel Object Oriented Specification Language' (POOSL). The simulation tool (SHESim) included facilitates verification of the behavior.
Several representations build up a formal model. Examples are a data model and an object class model, a collaboration model in the form of 'Message Flow Diagrams' (MFDs), a dynamic model in the form of 'Message Sequence Charts' (MSCs) and an architecture model.

The intended purpose of this study has been to make up an inventory of needs (SyRS) and to demonstrate the use of an object oriented modeling technique (SHE), as a potential way to both fundamentally and structurally specify and model a Maintenance Information Communication System.

This report shows an example of an approach to handling specification, modeling and design of a complex system.

Samenvatting

In dit rapport staan specificatie en modellering en een Communicatiesysteem voor Onderhoudsinformatie centraal.

Om de liberalisering van de elektriciteitsmarkt (sinds 1 januari 1999) te kunnen overleven, zullen energiebedrijven hun integrale kosten moeten drukken, waaronder onderhoudskosten, reparatiekosten en afschrijving.
Een manier om de kostenreductie te bereiken is door het onderhoudssysteem te verbeteren. 'Condition Based Maintenance' (CBM) and 'Reliability Centered Maintenance' (RCM) worden vaak in één adem genoemd als 'ToestandsAfhankelijk Onderhoud' (TAO) en zijn voorbeelden van 'Just-In-Time' Maintenance (JIT-M). Terwijl JIT-M enerzijds de onderhoudskosten drukt, wordt anderzijds een levensduurverlenging van apparatuur bereikt, hetgeen een aantal kostbare reparaties overbodig kan maken. Afschrijvingskosten zullen dalen als gevolg van het mogelijke uitstel van vervangingswerkzaamheden en nieuwe investeringen.

Om TAO (Toestandsafhankelijk Onderhoud) te plegen, zal er gedetailleerde informatie moeten zijn, die iets zegt over de toestand van betreffende componenten. Deze gegevens kunnen o.a. middels 'Remote Monitoring' (RM) worden verkregen. Een groot aantal bestaande diensten of bronnen kan hierdoor worden gedeeld op stationsniveau of centraal niveau, hetgeen de stap naar centraal onderhoudsbeheer vergemakkelijkt, gezien de hoge efficiëntie. Bovendien kunnen nieuwe diensten op systeemniveau de mogelijkheden van het systeem uitbreiden. Hoewel er veel voordelen van RM lijken te bestaan, verdienen o.a. veiligheid en snelheid (real-time) extra aandacht.

Aan het systeem worden eisen gesteld, welke zijn verzameld en vastgelegd in een 'System Requirements Specification' (SyRS), gebaseerd op een IEEE standaard. De SyRS is een hulpmiddel om de mogelijkheden en beperkingen van het systeem te analyseren, hoewel de beperkte tijd het opstellen van een complete specificatie niet heeft toegelaten. Tijdens de uiteindelijke ontwerpfase zullen beslissingen worden gemaakt op basis van de SyRS.

Naast specificatie, is ook modelvorming een belangrijk onderwerp in dit rapport. Er is gebruik gemaakt van een object georiënteerde ontwerpmethodiek, te weten 'Software/Hardware Engineering' (SHE). Het gedrag van het systeem wordt m.b.v. 'Parallel Object Oriented Specification Language' (POOSL) vastgelegd ('geprogrammeerd') in een model. Verificatie van het gedrag vindt plaats d.m.v. simulatie (SHESim).
Het formele model kent verschillende representaties, zoals bijvoorbeeld een data model en een model van objectklassen, een interactie model in de vorm van 'Message Flow Diagrams' (MFDs), een dynamisch model: 'Message Sequence Charts' (MSCs) en een architectuur model.

Het doel van dit onderzoek is om een inventarisatie te maken van behoeften (SyRS) en om het gebruik van een object georiënteerde modelleringstechniek (SHE) te demonstreren als een manier om een Communicatiesysteem voor Onderhouds-informatie zowel fundamenteel als structureel te kunnen analyseren specificeren en ontwerpen.

Dit rapport laat een voorbeeld zien van hoe specificatie, modellering van complexe systemen kan worden aangepakt.

Acknowledgements

At the end of my graduation project and the writing of this thesis, I would like to thank a number of people, who have supported me during this last year of my studies.

First of all, I would like to thank KEMA Nederland B.V. for their contribution and for giving me the opportunity to do my graduation research project in a business environment. It has given me a good impression of the atmosphere in a company. To mention a few people by their names: Dick van Aartrijk and Teus de Zwart, who put me in contact with KEMA, Jos Wetzer as my coach and advisor, and roommates Jan-Eldert Schieven and Casper Stoelhorst: our traditional '*Patatje Babi Pangang*' lunch breaks at Thursdays must certainly not go unmentioned!

Further, from the university's side, I thank Mr. Stevens for taking the responsibility of being my supervising professor. I would also like to thank Piet van der Putten for the effort and time he spent as my coach and advisor, in particular during the last two months.

And last but certainly not least: the home front.
I am greatly indebted to my parents, for they have always been a prop and stay to me. They have supported me both mentally and if necessary concerning content, during my entire education up to and including the final stretch.
I thank my sister, my in-laws and close friends, for their interest in my work.

Finally, special thanks go out to my girlfriend and confidante, Janet, who has had her hands full with me, particularly during the last few months. She always succeeds in putting heart into me and encouraging me at the right time.

Preface

This thesis is the result of my final project at the Eindhoven University of Technology (EUT).

As a part of the education program of the department of Electrical Engineering, the final project of at least six months, is preferably carried out in the industry, in cooperation with one of the department's research groups.

After a period of internal practical training, as well as the experience of a work placement at a university abroad, I decided to do my final project externally, at a company.

My interest in communications brought me to the Information and Communication Systems Group (ICS) at the university. My request to KEMA Nederland B.V. in Arnhem resulted into a project proposal that ICS has agreed with.
The research has been carried out at the business unit of Transmission and Distribution of Power (TDP), section Power Equipment Consultancy (PEC). One of their activities is the management of maintenance of power equipment.

We agreed on the project's subject, being datacommunications for remote on-line[1] monitoring on behalf of maintenance in the Dutch power system.
The intended purpose is to contribute to the specification and design of an adequate datacommunication system for maintenance information. This preliminary study serves as a basis for further investigations.

This report describes the specification of requirements, as well as the methodology for modeling and design. The approaches will be clarified by applying them to a part of the system.

[1] In this context, '*on-line*' means '*while in operation*'

Abbreviations

CBM	Condition Based Maintenance
CM	Corrective Maintenance
DGA	Dissolved Gas Analysis
EUT	Eindhoven University of Technology
TUE	Eindhoven University of Technology (Technische Universiteit Eindhoven)
EM	Energy Management
EMS	Energy Management System
END	Energy Not Delivered
EU	European Union
EHV	Extremely High Voltage
FMECA	Failure Mode and Effects and Criticality Analysis
FTP	File Transfer Protocol
FRA	Frequency Response Analysis
GIS	Graphic Information System
HV	High Voltage
HTTP	HyperText Transfer Protocol
Online	In operation
ICS	Information and Communication Systems
ICT	Information and Communication Technology
ISD	Instance Structure Diagram
IED	Intelligent Electronic Device
IEC	International
IP	Internet Protocol
IPv6	IP version 6
JIT-M	Just-In-Time Maintenance
KPN	Koninklijke PTT Nederland
LMC	Local Management Center
Onsite	Locally
MSO	LongRange Research & Development Programme (Meerjarenplan Studie en Onderzoek)
LV	Low Voltage
MICS	Maintenance Information Communication System
MM	Maintenance Management
MMS	Maintenance Management System
MV	Medium Voltage
MFD	Message Flow Diagram
MSC	Message Sequence Chart
NMC	Network Management Center
Remote	Not located at the substation
OCD	Object Class Diagram
OO	Object-Oriented
Offline	Out of operation
POOSL	Parallel Object Oriented Specification Language
PD	Partial Discharge
PeM	Periodical Maintenance
PEC	Power Equipment Consultancy
PrM	Preventive Maintenance
QoS	Quality of Service
RCM	Reliability Centered Maintenance
RM	Remote Monitoring
R&D	Research and Development

SSL	Secure Socket Layer
SHESim	SHE Simulator
SRS	Software Requirements Specification
SHE	Software/Hardware Engineering
SyRS	System Requirement Specification
TCD	Tap Changer Diagnostics
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
TBM	Time-Based Maintenance
TAO	Toestandsafhankelijk Onderhoud
TDS	Transformer Diagnostics System
TDP	Transmission and Distribution of Power
TDP	Transmission and Distribution of Power
TCP	Transmission Control Protocol
T&D	Transport and Distribution
UFM	Unified Formal Model
VDE	Virtual Distribution Equipment
VPN	Virtual Private Network

Contents

Chapter 1. General Introduction

This chapter is an introduction to the project described in this report. Questions about the intended purpose, what to achieve, and how, will be answered here.
The expected results of my contribution to a more extensive project are given, in order to define boundaries of my graduation project.
Furthermore, the potentially interested parties are mentioned briefly.
Finally, the outline of the entire report is given.

1. General Introduction

Compared to the fast growing world of information management and communication, the world of electrical power delivery engineering seems to be relatively stable. However, new developments in the Information and Communication Technology (ICT) increasingly influence the electrical utilities.
Although from a technological point of view the field of electrical energy engineering is indeed developing with a moderate speed, big changes in the sector of energy distribution are taking place. The growing competition between energy companies, caused by the liberalization of the energy market, emphasizes the need for keeping up with the newest technologies. As a result, the energy companies will have to offer their services at higher quality and at lower prices.
Improving the maintenance system is one of the ways to achieve these goals. In addition, the ICT (Information and Communication Technology) can be used as a tool to realize them.

1.1 The Need for a MICS

The mentioned liberalization forces companies to compete. In order to keep their prices low, maintenance costs are being reduced, by moving over from Time-Based Maintenance (TBM) to Just-In-Time Maintenance (JIT-M). Unnecessary expensive maintenance will be avoided; a company only maintains when necessary. Condition Based Maintenance (CBM) and Reliability Centered Maintenance (RCM) are JIT-M strategies. Maintenance can be scheduled on the basis of condition information or its possible consequences (reliability), with delayed or intensified inspection rates depending on the results.
Starting from the system's reliability, which has been laid down during the design phase, the equipment's condition plays an important role for the implementation of CBM. The question is how a component's condition is determined, and how it is processed. There are many developments to determine the condition and/or the reliability of certain devices. In the future, investigations in this field will only be intensified. The current reliability of the system is based on this information.

An advanced communication system to transport correct information at an appropriate time, from and to specific locations will be necessary in order to meet the requirements.

1.2 Approach

Methodologies for both specification and design play an important role during (and after) this project. Traditionally, specification, design and implementation are subsequent phases in the process of system development. It may take a large number of iterations before the system is optimized and ready for being used.

To reduce the number of iterations during the design phase, another approach is required. A more flexible methodology with feedback and interaction during the design phase, between all those concerned, prevents minor modifications of the

specification from having large consequences on the system's design and implementation.

One of the main issues in the project is drawing up requirements, with as a result a well-organized specification document. Specifications are frequently used as a way to communicate between people from different disciplines. Therefore, it is highly important that the document is unambiguous and cannot be misinterpreted by its different audiences.
An interactive process between customer and the technical community involved leads to a set of requirements that define the system's capabilities and behavior in several predefined situations (scenarios).
A System Requirements Specification (SyRS) is a dynamically changing document. Because of the interactive way of deduction and evaluation, requirements will constantly be subjected to adaptations during the design phase of a system.
An IEEE[1] template lays the foundation of an SyRS (System Requirements Specification) in this project. The most important argument for this decision has been the benefits of standardization of (future) specification documents, preferably, based on the results of comprehensive research by a well-established organization.

Many companies move over to an Object Oriented (OO) modeling approach for system modeling. OO-modeling aims at 're-usability'.
At a higher level, the OO-analysis methodology means re-usability of the way of thinking: the same approach can be followed for the design of similar complex systems.
On a more detailed level, it allows drawing multiple instances from generic object classes. Each instance has the same structure, but its own state. In other words, the object's class can be seen as a building brick that is re-used.
When a specific structure of objects returns several times, object instances can be grouped together into an object cluster. The definition and re-usage of object clusters retain the clear organization at a higher abstraction level.

Most of the object-oriented methodologies do only apply to software systems. However, the Software/Hardware Engineering-methodology (SHE) – developed by EUT[2] – handles both hardware and software systems, which explains its use in this project.

SHE supports a formal description of the behavior of objects in Parallel Object Oriented Specification Language (POOSL). A simulator runs the formal specification and visualizes communications between different objects or clusters of objects. In this manner, the system's specification can be checked for errors and imperfections. The power of the simulator extends to the function of a communication device itself. People from different disciplines are able to see the system's behavior at once, and react by giving feedback. It stimulates the interactive way design should be, and makes it more accessible.

1.3 Problem Boundaries

This preliminary study investigates an approach to the development of a MICS (Maintenance Information Communication System). Although some parts are discussed in detail, this preliminary study serves as an exploration of the problem

[1] IEEE: The Institute of Electrical and Electronics Engineers, Inc.
[2] EUT: The Eindhoven University of Technology

domain, using a concise specification and model. The exploration can be useful for continuation of the project.
A more important goal is to demonstrate the use of structured analysis, specification, modeling and design as a solid basis for the development of complex systems like this one.
After sketching the context, the system's boundaries have become clear. One of the intended purposes is to show the required interaction with other systems across the system's boundaries. Secondly, the project concentrates on the information flow related to a part of the system in detail, namely an intelligent substation transformer. In this way, communications will be shown on both a global and a detailed level.

1.4 Stakeholders

This section mentions the parties that can or will be interested in the results of this preliminary study ('stakeholders')[1]. All parties concerned should be involved for the continuation of this project. Stakeholders are the industries with (electrical) infrastructures, manufacturers of equipment and KEMA.

The first potentially interested party is the (electrical) utilities and other industries with (electrical) infrastructures.
For organizations that do not make use of a CBM (Condition Based Maintenance) strategy, the system makes the step towards CBM more accessible, because of its many advanced and advantageous capabilities.
Organizations that do already use CBM can add functionality by using Remote Monitoring (RM).
The system's influence will not be restricted to operations. The planning department, for example, can also count on an improvement of management, due to an extension of the term in which replacements and investments can be scheduled. New machinery need not be over-dimensioned as today; the minimum requirements can be met more tightly.

In the second place, the manufacturer of equipment for the utilities might be interested in the results of this study (and its continuation). Information on specific characteristics, imperfections or even failures of a particular type of equipment (or its manufacturer) will become available. The information is based on the entire population, instead of single incidents, which makes it more important to the manufacturer, because it can be taken into consideration for future equipment development.

And last but not least, the results of this study will be useful to KEMA Nederland BV, who initiated the project in the first place. The generic part of the system can be specified and designed by KEMA. The generic system would then be one of KEMA's products that can be customized for any individual case.
Subsequently, KEMA can offer consultation along with the generic system. So KEMA and her customers cooperate and work towards a specific implementation.

As the Dutch energy distribution companies have combined their Research and Development (R&D) efforts in the field of electricity, gas and heat supplies into the Long Range Research & Development Program (MSO[2]) of EnergieNed, it would be

[1] Not to be confused with 'stockholders' - Stakeholders are a collection of potentially interested parties that may profit from the results of this report
[2] MSO (in Dutch): Meerjarenplan Studie en Onderzoek

sensible to carry out this research via MSO. The results of the MSO will then be available to all contributing EnergieNed members[1].
In case of a fusion between (distribution) companies, the connection of independently developed proprietary systems would bring nothing but misery (e.g. by losing functionality). These problems can be avoided by developing together merely one universal system.

Additionally, it can be said that the benefit of the (chosen) generic approach is the ability to embed the system approach into different organizations. Utilization isn't necessarily bounded to the energy transport and distribution sector, but can be applied wherever remote maintenance is profitable.

1.5 Arrangement of Chapters

Chapter 2 examines the system's context, the Dutch power system. Both technical and non-technical aspects are brought up.

Background material on 'remote monitoring' and its relation to the subject is given in Chapter 3. The importance of RM (Remote Monitoring) for energy companies to survive the future is explained.

Chapter 4 discusses the approach for specifying this complex system. Both requirements capturing process and the SHE (Software/Hardware Engineering) methodology will be explained, as an approach for specifying the behavior of the system using requirements, models and simulations.

An example of users' requirements for a remote maintenance system follows in Chapter 5.

A simplified RM system model is described in Chapter 6. Several representations build up a formal model of the system. The system's desired behavior, in particular regarding communications, is defined in various situations.

Finally, conclusions are drawn in Chapter 7, followed by recommendations for continuation, utilizing the results, achieved during this project.

[1] Provided that the MSO will continue in one form or another.

Chapter 2. The Dutch Power System

According to the previous chapter, the intended audience for this project consists of people with many different interests. For those who are not familiar with the Dutch power system, this chapter provides for background information.

2. The Dutch Power System

This chapter provides an insight in the system's context.
The first section describes the Dutch power system in general. It contains hierarchical information on the classification into multiple voltage levels. The various sectors the electrical utilities consist of will be discussed briefly. In general, four parties are mentioned, namely production, transport, distribution and consumer, of which transport and distribution are the most interesting ones for this project.
Section 2.2 discusses the physical structure of the electricity grid, managed by an electricity company. Also discussed are several important elements in the sub network, i.e. connections (cables and lines) and substations. Failure Mode Effects and Criticality Analysis (FMECA) points out a number of components that should be subjected to a more detailed inspection. One of those, the energy transformer, will be used as an example throughout this report.
The management of distribution networks is explained in section 2.3.
Section 2.4 discusses different strategies of maintenance, in particular Condition Based Maintenance.
Finally, section 2.5 brings up a few words about liberalization of the energy market in the Netherlands.

2.1 System Context

This section provides for some background information on the Dutch power system. The function of the electrical utilities is to provide their customers with electrical energy. To get the required energy from the source (production) to the destination (customer), and to connect all production entities, an electricity network is needed (see Figure 2.1). Our major concern is the network part (transport and distribution).

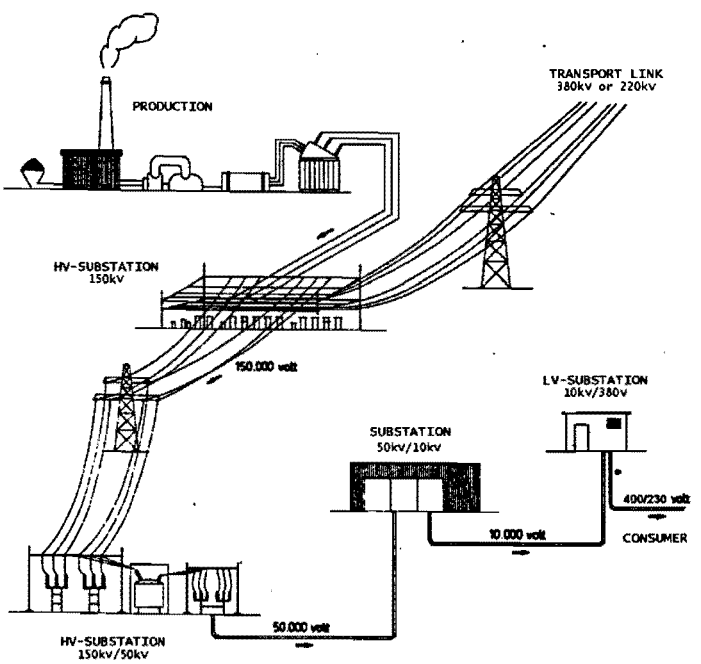


Figure 2.1 Electricity from production to customer

Energy networks can be distinguished either by their function or by their operating voltage.

Different functions:

A network is called *transport* or *transmission* network, when it transports large quantities of energy either to the distribution centers or between large and important substations. Transport links usually cover long distances.
In some literature, the term ***sub*** *transmission* network is used to differentiate between transport functionality on Extremely High Voltage (EHV) level (see Table 2.1) and below (i.e. High Voltage (HV) down to Low Voltage (LV)). This sub transmission network connects the main transmission network[1] (between main stations[2]) to the main distribution substations.

A network is called distribution network, when its primary function is to divide an amount of electrical energy into numerous small portions.

Different voltages:

Next to the division on the basis of function, there exists a classification into voltage levels.
The electricity voltages that are used in the Netherlands vary from 400 V (coupled voltage) on customer level to 380 kV in the main network (see Figure 2.2).

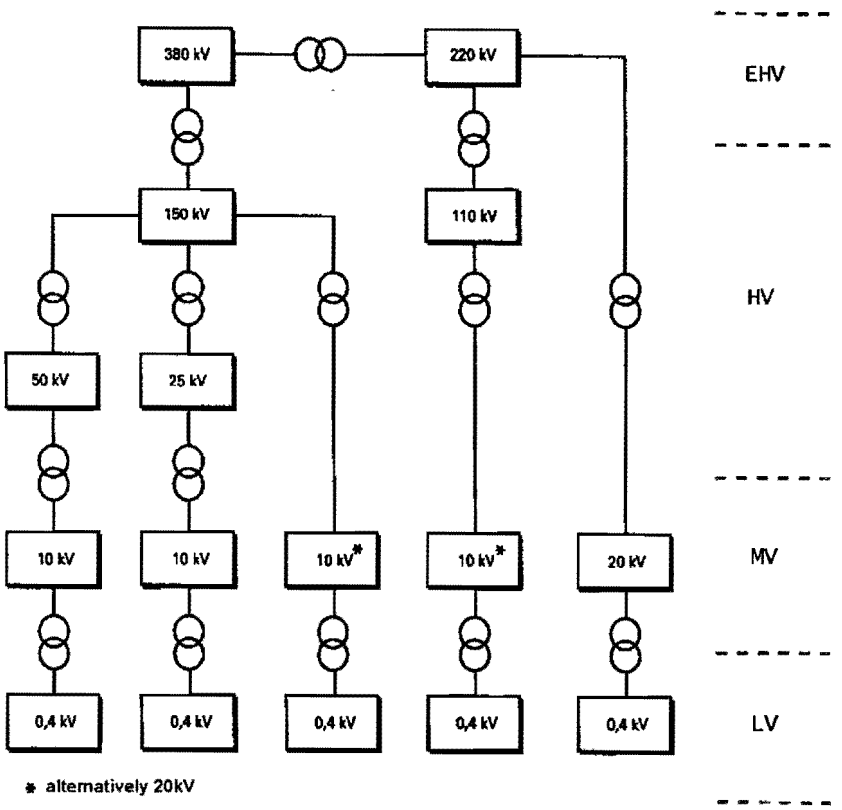


Figure 2.2 Commonly used voltage levels in the Dutch power system

[1] In Dutch: Koppelnetwerk
[2] In Dutch: Koppelstations

The values of the voltage ranges (Extremely High, High. Medium and Low Voltage, or EHV, HV, MV and LV, respectively) do not fully correspond to one of the four configurations of the IEC[1] recommendations (Table 2.1), but we have chosen the configuration of the last column. In this way, functionally different voltage levels are assigned to different categories.

The main transport network operates at EHV level. Sub transport networks use voltages between 10 and 150 kV. Networks up to 10 kV function as distribution networks.

There are three kinds of electricity companies:
1. production companies
2. one main transport company (managed by TenneT[2])
3. distribution companies

Although the latter (3) may suggest differently, 'distribution companies' also manage sub transport networks, up to 150 kV. Apart from TenneT, which manages the main transport network on EHV-level, there are no separate transport companies below 220 kV. This explains why the term 'transport and distribution companies' is hardly ever used, although it would save some confusion.
The function of a production company (1) should be clear.

Table 2.1 Voltage levels categories

voltage level	configuration no. 1	configuration no. 2	configuration no. 3	configuration no. 4	Dutch configuration
EHV	765 400/500 220	765 400/500 220	765 400/500 230	765 400/500 230	380 220
HV	110/132 66	110/132 66	115/138 69	115/138 69	110/150 25/50
MV	33 22 11	35 20 10	33 22 11	35 20 10	20 12.5 10 3
LV	0.4	0.4	0.4	0.4	0.4

Figure 2.3, on the next page, gives an overview of the higher voltage level infrastructure in the Netherlands.

Different network structures (topologies) will be discussed in the next section.

[1] IEC: International Electrical Committee
[2] TenneT: Main Transmission System Operator in the Netherlands

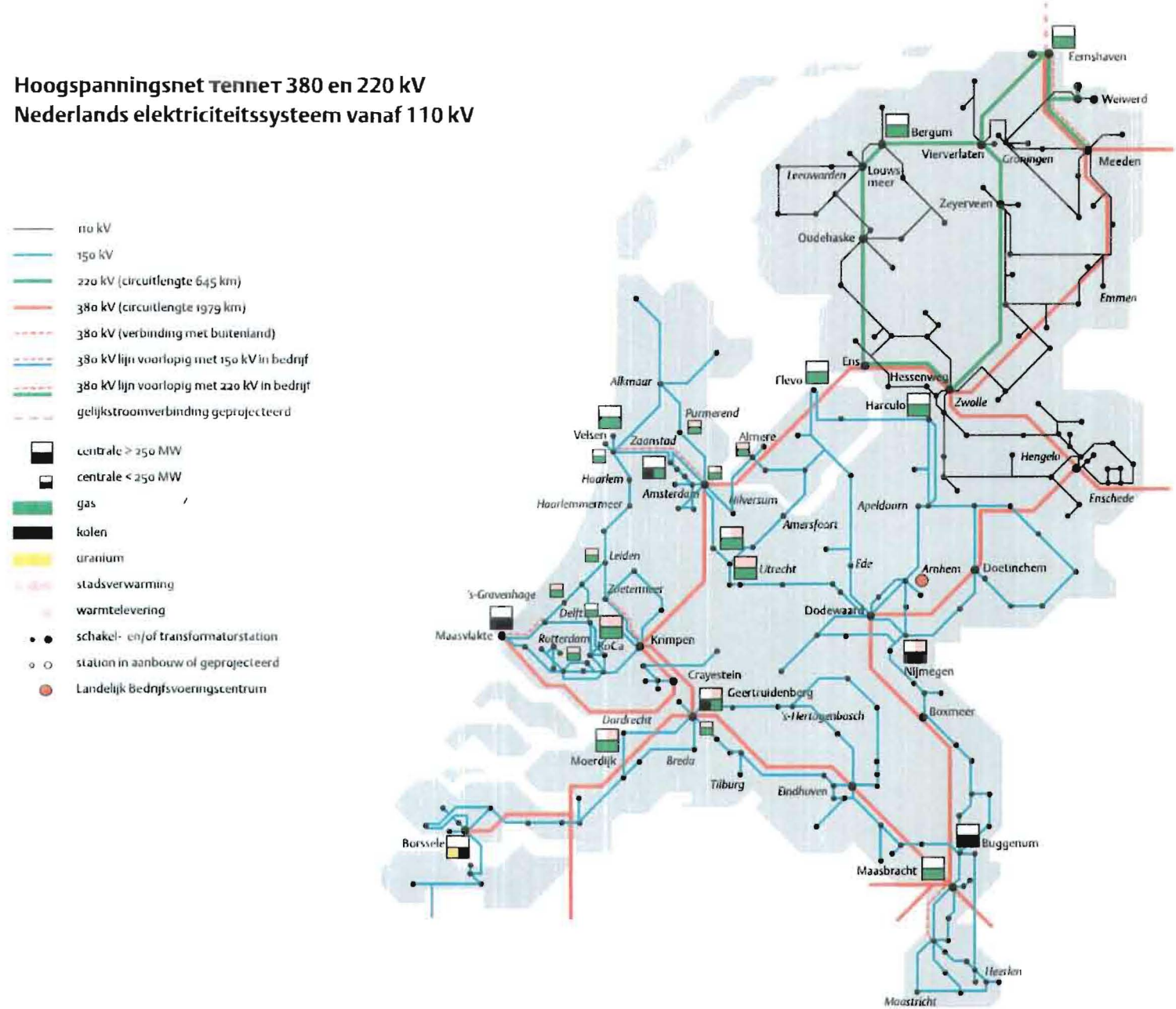


Figure 2.3 The Dutch EHV and HV electricity infrastructure

2.2 Physical Network Structures

This section describes the construction of networks that are managed by the electrical energy distribution companies. Before the physical structures will be explained, the following should be mentioned.

First of all, neither the main transport network, nor electricity production units will be discussed further in this section. This simplification allows us to dissociate from irrelevant parts. The network is seen as a black box that connects to the main transport network on the one hand and to customers on the other hand (Figure 2.4).

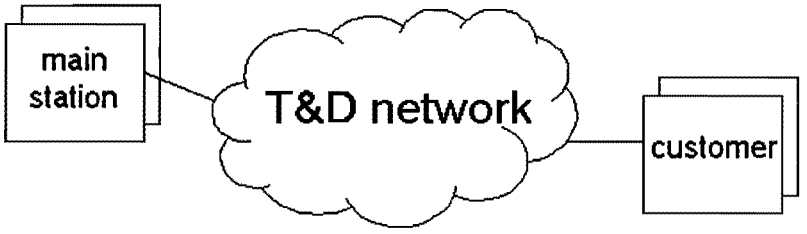


Figure 2.4 T&D network 'black box', owned by a distribution company

All Transport and Distribution (T&D) networks provide a certain area, usually a geographically determined region, with electricity.
The levels, into which each T&D network can be subdivided, are based on the voltages from Table 2.1 in the previous section. This work focuses on the highest level inside: the HV level (110 kV and 150 kV transport network).

Sections 2.2.1 through 2.2.4 handle the network's physical structure.

2.2.1 Transmission lines

Any physical (sub)network or infrastructure consists of several *nodes* that are *connected* to other ones in a certain way.
Either transmission lines or cables are applied to transport electrical energy from and to substations. The higher voltage transmission networks commonly use transmission lines through the air; distribution networks below 10 kV use underground cables. However, in an exceptional case, also transmission networks (up to 380kV) use cables instead of lines.

The pylons that hold the transmission lines are dedicated to a specific voltage level. Figure 2.5 on the next page shows some examples.
A number of lightning conductors (1-3) protect the phase conductor wires from being hit by lightning and can be found above the three wires. Some of these lightning conductors contain optical fiber, which is used for the point-to-point communication between substation and network management center. Except from urgent protection purposes, substation cannot communicate with each other.

Some 'smart' cables also contain optical fibers, for the purpose of temperature measurements. Although the use of these fibers would be ideal for datacommunications, still a lot research has to be done on its influences on the temperature measurements, vice versa. However, most smart cables contain more

than only one single optical fiber, which makes communication via the HV-lines very attractive.

To ensure the continuity of electricity delivery and to prevent lines from being overloaded, double circuit connections add redundancy to the (E)HV-transport networks.

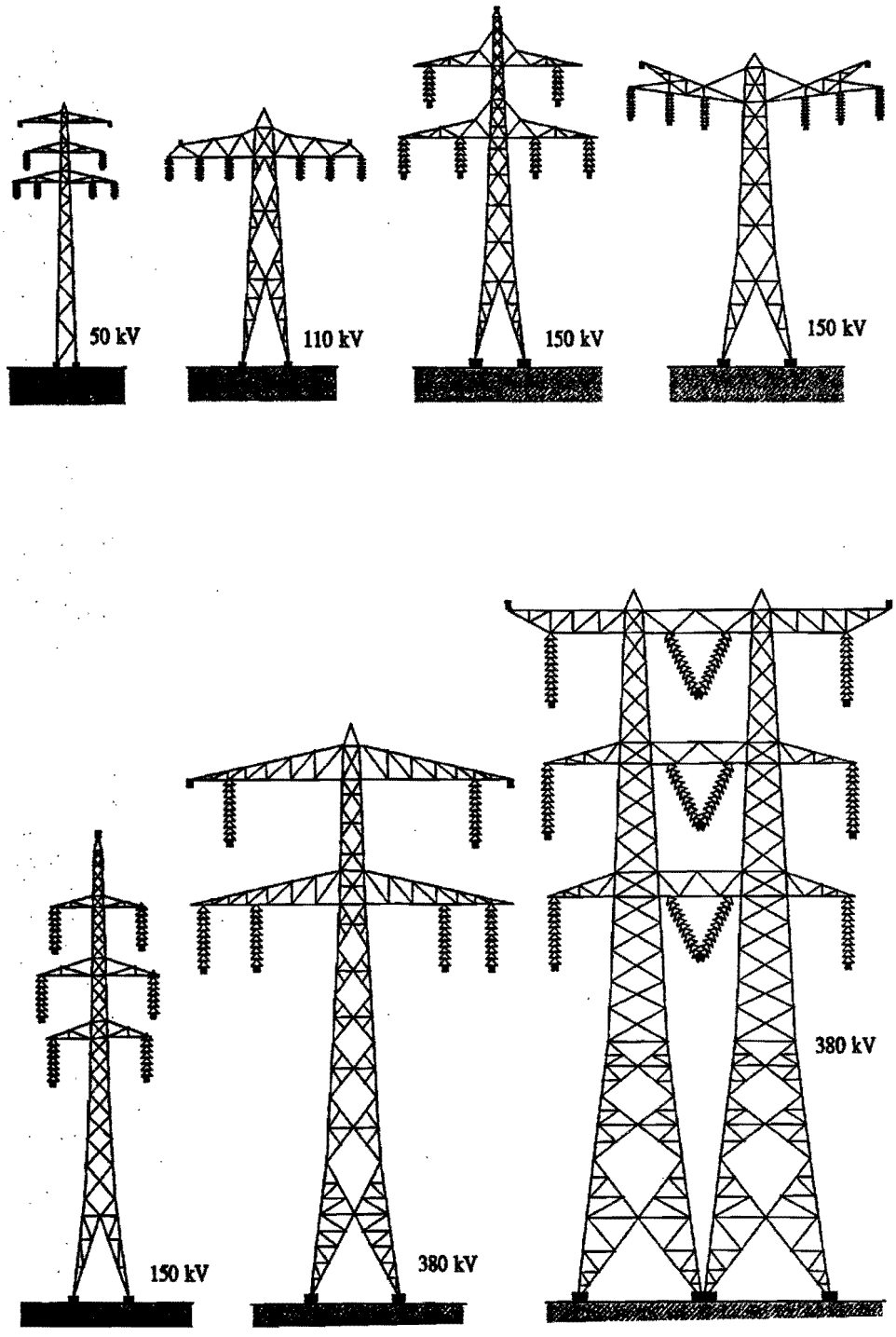


Figure 2.5 Several pylons at different voltage levels

To avoid possible problems with the definitions and terms that are used in this report, an explanation follows:

Each pair of three phase conductors (**transmission lines**) will be called a **field**. A **connection** between substations typically consists of two fields. In just a few cases, a connection between substations consist of one field; sometimes even more than two fields are used to build up a connection.

2.2.2 Topologies

Beside the question whether either single or double circuits are used, the network can be configured in several ways. Each network structure (topology) has its own benefits, regarding management, security, availability and maintenance.

Three basic network structures are shown in Figure 2.6: *star topology*, *ring network* and *a meshed network*.

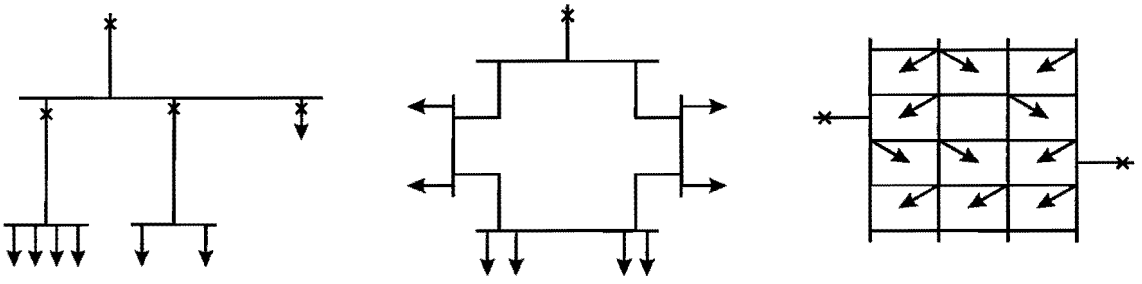


Figure 2.6 Basic network topologies: ***star*** *(left),* ***ring*** *(middle) and* ***meshed*** *(right)*

The greatest benefit of the star topology is its simplicity. Star networks are not complex and easy to manage. However, a great part of the network might be unavailable in case of a failure or in case of maintenance.
Additional connections between substations will increase the availability.

Each substation in a ring network can be reached via two different routes. This means that in case of a single connection failure, all stations can still be supplied. Another benefit of rings is load distribution. When a certain substation consumes relatively much power, other stations can be supplied via the other route.

Even better load distribution can be obtained by using a meshed network. Each station can then be fed via more than two routes.

The drawback of an increased availability is the increase of complexity, and thus manageability. Existing networks use combinations of these topologies.
HV networks mostly use ring structures, with a few single branches outwards. Implementing several parallel connections provides the security of HV networks. A number of connections to the main 220 kV and 380 kV transport network supplies the HV network.
MV *transport* networks generally make use of ring networks or several parallel connections; MV *distribution* networks use both star topology and ring structures that can be disconnected at different locations. The meshed network topology can be found in LV networks.

2.2.3 Substations

Substations operate as nodes in transport and distribution networks. They connect networks at different levels and connect several segments of transmission lines. This section describes the substation's function and construction in a more specific way.

Function

Substations can be distinguished to their functions. In general, there are two different functions: connecting networks of different voltages and distributing electrical power to different areas.

The main function of substations in the higher voltage level layers of the electricity grid is supplying the layer below. Transformers within the substations take care of voltage level transformations.

The other function of substations is the distribution of electrical energy. Distribution substations can be found in the lower voltage level networks (up to MV: 10 kV) and their primary function is branching off multiple network segments from the supplying segment(s).

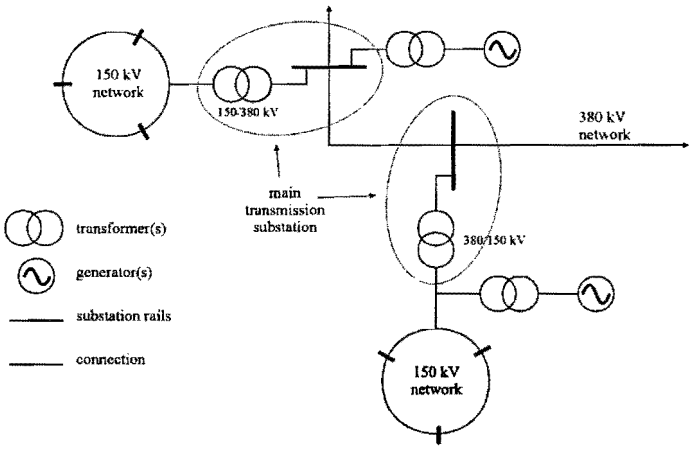


Figure 2.7 Different networks interconnected by substations

Figure 2.7 shows an example of the connection of several network layers by the transformers of substations. Transformers are used to convert the EHV-level of 380 kV to 150 kV of the HV-network, and can be found within main transmission substations. This figure also includes power generators, feeding either the HV-networks or the EHV-network via a transformer.

Construction

Switching fields

Most of the HV substations in the Netherlands are constructed in open-air, although the newer substations have closed constructions with SF_6 gas isolation. A few benefits of this newer technique are compactness, lower maintenance costs, increased safety and their lower vulnerability to environmental influences (e.g. air pollution and precipitation).

Major components are the switches, rails and transformers. Figure 2.8 on the next page shows a 'switching field', i.e. a connection of a certain field to the rail systems (1). Isolating switches (2) can segregate a field (7) from the rail systems. For continuity reasons, a second rail system is present, which can be used as a backup. The power switch (3) is able to switch large currents, especially in case of a short circuit and other fault situations. The isolation switches, however, only switch at zero-current.
Grounding switches (4) ensure a zero-voltage level in case of maintenance, purely for safety reasons. Transformers (5) and (6) are used for current and voltage measurements, respectively.
The more modern substations are equipped with Gas Isolated Switchgear (GIS), which means that all parts of the switching field of Figure 2.8 are enclosed by a gas filled cylinder.

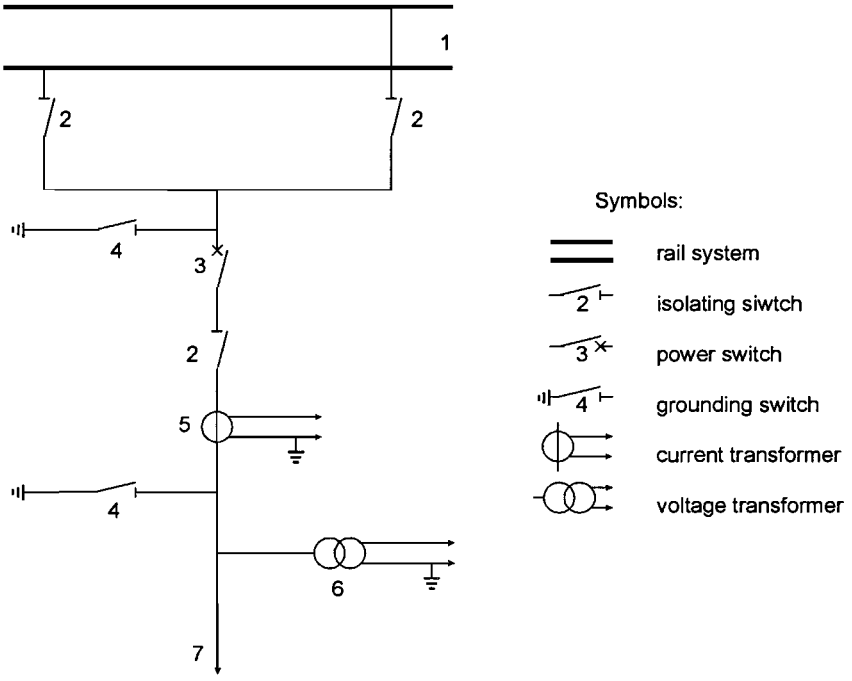


Figure 2.8 Switching field of a substation

Transformers that feed lower voltage networks are connected to (7). Other fields are part of the connection to another substation in the same network level.

Secondary Equipment
Next to the primary (power) equipment, the secondary system that is present in all HV substations takes care of measurements, control and security. Critical measurement data are transported to the distribution company's management center. The secondary system is also capable of reacting autonomously on failures by taking immediate actions.
Usually an operating crew is present or available for 24 hours a day in every HV-substations and in some more important MV-substations. Information on network management can be found in section 2.3.

As in chapter 6 the choice has to be made on which component to maintain in the example model (transformers), a more in depth description on power transformers can be found in the next section.

2.2.4 Power Transformers

According to the definition stated in IEC 76-1, a transformer is *'a static piece of apparatus with two or more windings which, by electromagnetic induction, transforms a system of alternating voltage and current into another system of voltage and current usually of different values and at the same frequency for the purpose of transporting electrical power'.*

Next to the measurement transformers, mentioned in the previous section, there are several types of transformers, like power transformers.

Figure 2.9 shows an impression of the transformer's functions, subsystems and components.

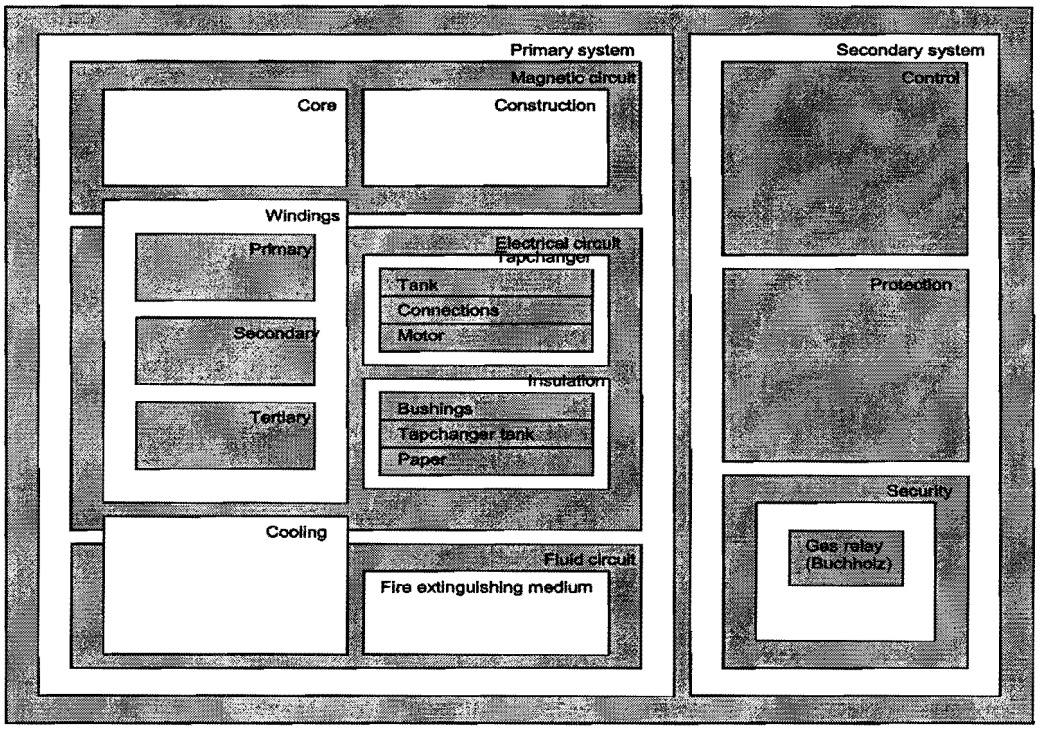


Figure 2.9 Transformer's contents

First of all, a transformer consists of a **primary system**, containing all power equipment and a **secondary system** for control, protection and security.
The primary system can be subdivided into a magnetic, an electrical and a fluid circuit, each of them containing a number of subsystems or components.
Actually, an overlap exists between the three circuits, e.g. the winding are part of both the magnetic and electrical circuit; the oil serves both a thermal function (cooling fluid circuit), and a dielectric one (electrical insulation circuit).
The secondary system is connected to the secondary equipment of the substation.

2.3 Network Management

Definition: *"Network management is the way in which the network concerned takes care of the transport of electrical power from supplier to consumer in both normal operation mode and trouble situations."*

The network supervisor at the head of each distribution company is responsible for the correct functioning of his network. This means providing a certain 'Quality of Service' (QoS) which their customers pay for.

Guaranteeing the availability of a certain amount of electrical power means that the network manager has to make provision for possible trouble. In that case, the offered services should not be interrupted, or the interruption should be reduced to a minimum.

2.3.1 Network Management Center

The Network Management Center (NMC) of a distribution company is the network manager's home base. Global operational information on the current situation is collected here. Network operators can monitor several characteristic parameters, like the current load, voltage or current on a particular network segment.

Functions:

A lot of the management functions are concentrated in the NMC. For this project, the two most important are 'Energy Flow' and 'Maintenance Management'.

Energy Management
One of the functions of the NMC is controlling the network's primary purpose: transport and distribution of electrical energy. Measurement results of process parameters of components in the HV-network are collected by the Energy Management System (EMS). Load flow calculations can be made on the basis of the information provided by this EMS, which allows routing the electrical energy in either the most economical way, in the most efficient way, or in the most reliable way.

In case of an error, the components concerned take immediate action, and report it to the NMC. Operators can decide what strategy to follow in order to correct the problem and to prevent these problems in the future.
Directly after the occurrence of a critical situation, the network manager may decide to change or adjust the energy flow within the network, in order to prevent other network segments from being afflicted.

Maintenance Management
Another function of the NMC (Network Management Center) is the management of maintenance. Strategic decisions are made on when to maintain equipment, when to make new investments, when to replace what equipment, in other words: scheduling maintenance.
The information on which these decisions are based is obtained via local inspections and measurements. Some information analysis work is subcontracted to an external party, as well as the maintenance itself.

A future MICS (Maintenance Information Communication System) can be a useful enhancement of Maintenance Management, by adding functionality, and by giving the different players additional autonomy.
Not only the NMC will initiate maintenance in the future, but also on the initiative of the equipment concerned itself, when a need for maintenance is signaled.

Other
Several departments within the NMC carry out a lot of other functions.
Examples are the financial department, customer services, billing, construction, business, etc.
The relation with the energy and maintenance management exists, but is not described further in this work.

2.4 Maintenance Strategy

The introduction has already mentioned the need for enhancing the maintenance system. Failure analysis addresses which parts of a device are sensitive to failures, and by what causes. One of the major causes of failures turns out to be bad or insufficient maintenance [52].
Liberalization requires control and reduction of costs, which, amongst others, can be achieved by adapting the maintenance strategy. This section discusses different methods of maintenance, in particular Condition Based Maintenance.

Figure 2.10 (often referred to as 'bathtub curve') shows the relation between failure rate of a device and its age.

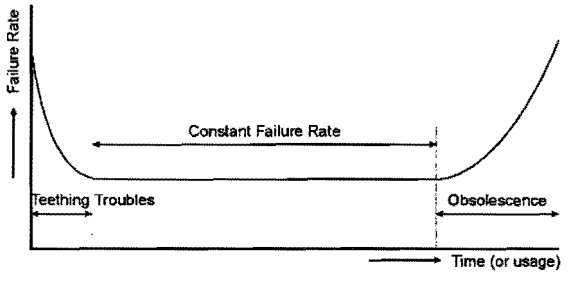


Figure 2.10 'Bathtub' curve

The initially high failure rate of new machinery (often suffering from teething troubles) decreases in time or usage. On the contrary, the rate of failures due to obsolescence increases in time or usage. The resulting bathtub-shaped curve shows a constant and lower failure rate in between.

With a well-thought maintenance strategy, a reduction of failures and costs can be achieved: *Teething troubles* of new equipment can be nipped to the bud.
Unnecessary maintenance costs in a period of *constant failure rate* can be avoided by maintaining less often, depending on the devices' actual maintenance needs. In addition, the extend of the equipment's life span (as a result of better maintenance) effects a drop of failures due to *obsolescence*.
It is important to find the right optimum between costs related to maintenance and costs related to a reduced reliability as a result of insufficient maintenance.

Figure 2.11 illustrates the relation between costs and reliability. An optimum can be found in the resulting curve of the total costs.

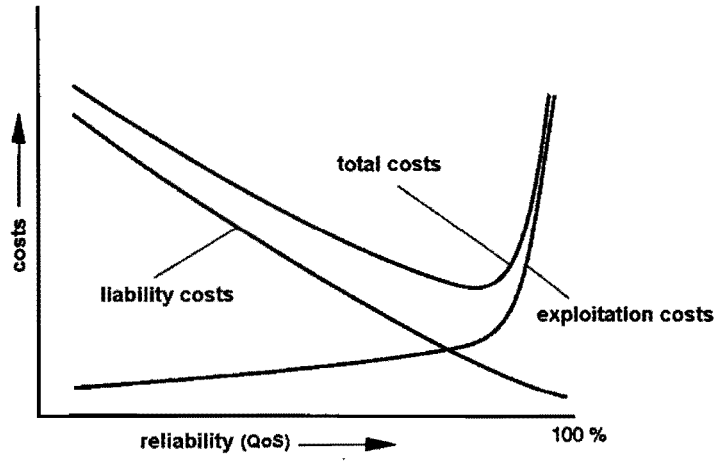


Figure 2.11 Costs vs. reliability

On the basis of this relation, a Network Management might decide to provide a high Quality of Service (QoS). The exploitation costs will increase as the offered reliability is raised.
To retain the chosen ratio of costs to reliability, the management in charge of maintenance will have to make the decision, whether to keep older equipment in operation (lower reliability), or to replace certain older parts with newer ones (higher maintenance costs).

In general, two different maintenance strategies can be distinguished, namely 'corrective' respectively 'preventive' maintenance.

2.4.1 Corrective Maintenance

Corrective maintenance means repair, after a problem has occurred. The total costs of repair and possible liability depend on the voltage level and the function of the component, but may be high. Corrective maintenance is very difficult to predict or plan. This is the reason why it is only used for cheap components without a strategic importance.

2.4.2 Preventive Maintenance

This method is a more elegant way of saving costs and preserving one's equipment. Generally, a maintenance schedule is set up, in order to prevent the equipment from failing.

Preventive maintenance can be subdivided into *periodical* and *condition based* maintenance.

Periodical maintenance

Maintenance will be carried out after a certain period. This may be a period of either time (TBM – Time Based Maintenance), usage, or any other countable quantity. For electricity networks, a number of quantities can be used, depending on the component (e.g. the number of actions for a switch, or the product of load and duration for transformers).
The major drawback of Periodical Maintenance (PeM) is the abundance of unnecessary service jobs. Sometimes the maintenance will not be necessary at all, while at another time the schedule should actually be intensified.

The extra maintenance may not completely justify the benefits in terms of preventing errors. Some people even claim that about 80% of the PeM costs could actually be saved.

Condition Based Maintenance

If it would somehow be possible to determine the condition of equipment, an (energy) company would merely maintain when strictly necessary. The frequency can be adapted: delaying non-critical service and intensifying maintenance when necessary. This strategy takes into account the equipment's need of maintenance, which works both ways. On the one hand, maintenance costs will remain at an acceptable level. On the other hand, the equipment's life span will be extended, which allows the delay of investments, while keeping quality of service.

RCM (Reliability Centered Maintenance) is a type of CBM in which decisions are made on the basis of expected reliability, instead of 'condition'. Although there is a slight difference (CBM represents the condition at the time of measurement; RCM looks forward to the consequences of the current conditions), the term 'CBM' often refers to both.

Table 2.2 summarizes the mentioned maintenance methods. While section 2.4.3 goes into the details of the CBM structure, additional information on several other forms can be found in [7].

Table 2.2 Overview of maintenance strategies

Maintenance	corrective (CM)	repair	
	preventive (PrM)	periodical (PeM)	time
			time-usage
			quantity-usage
		condition based (CBM)	condition
			reliability (RCM)

An optimal maintenance strategy is built up by a combination of methods, e.g. applying CBM to components with a strategic function, and applying TBM or CM (Corrective Maintenance) to other, non-strategic, ones.

Which approach should be taken depends on other factors as well. If maintenance or immediate replacement of an older piece of equipment is more expensive than letting

it run to failure and replace it, with all the involved consequences, a company may choose the latter option.

2.4.3 Condition Based Maintenance structure

Figure 2.12 compares PeM (Periodical Maintenance) to CBM. It briefly visualizes the essential differences.

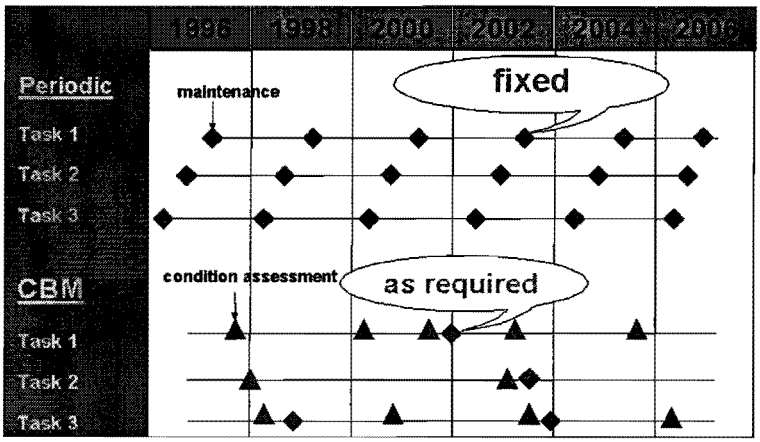


Figure 2.12 Periodic Maintenance vs. CBM

The realization of CBM includes several aspects, which will be mentioned briefly. The first step is the development of maintenance concepts that allow CBM. Which concept best suits a specific situation depends on economical and technical assessments. Condition assessment (a profession on its own) relates diagnostic information to technical and economical values (Figure 2.13).

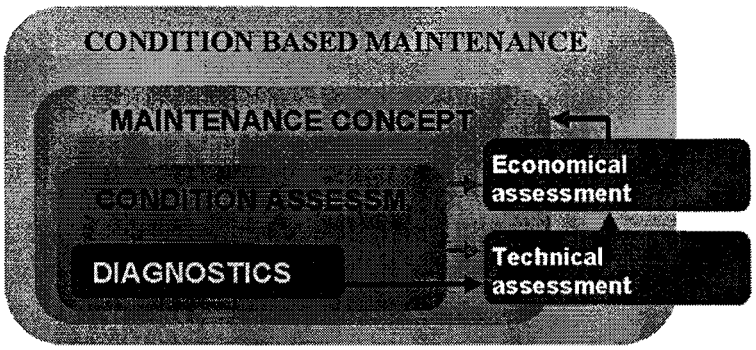


Figure 2.13 CBM structure

Diagnoses are based on information on the component. Measurements and diagnosis provide this information (next to failure reports and operational information). Several measuring techniques can be identified (periodically, continuously or incidentally). Some measurements can be carried out while the component is in operation ('on-line'); others require the component to be out of operation (offline measurements). A third option is to monitor either on-site (locally) or remotely. This project explores the remote variant.

Another subject will be the condition assessment. The question is how to determine the need for repair or overhaul on the basis of measurements. The needs can be classified, as some measures need to be taken immediately, while other ones can wait. In case of a suspect condition, the measuring frequency can be intensified.

To determine a transformer's condition, a number of monitoring tools has been developed along with diagnostic tools. Some of them can be used online (e.g. 'Dissolved Gas Analysis' (DGA) or 'Partial Discharge' (PD) measurement), some require the equipment to be offline (e.g. 'Frequency Response Analysis' (FRA), Tap Changer Diagnostics (TCD) and tan(δ) measurement.
More information on these diagnostic tools is available at KEMA.

Monitoring could also be an aid in locating a possible (future) failure, its cause and its stage.
CBM implies other fields of study, like e.g. designing CBM concepts on a component level, and developing diagnostics, including new measuring techniques.

An interested reader can find some additional information in one of KEMA's internal documents [49] and [50].

2.5 Liberalization

The liberalization slowly advances in a sector that was dominated for years by enterprises owned by the governments, also in the Netherlands. The new Electricity Act[1] has recently been accepted in the Dutch Parliament, the same is happening with the new Gas Act[2]. The same process occurred earlier with the Telecommunication sector, where KPN's[3] monopoly position has been broken.

The liberalization of the electricity sector is driven by the European Union (EU). According to guidelines of the EU for the internal market, the national, protected and fragmented electrical energy market has to be opened for competition. In the Netherlands this process is slowly progressing, especially compared to the United Kingdom and Scandinavia, who privatized their utilities (driven by a political choice) even before the EU guidelines were formulated.

A specific time-trajectory is set up. With the new Electricity Act and Gas Act, new producers can offer their products to the energy market. Consumers are allowed to select their preferred supplier; for the big consumers this is from the year 1999, for the medium sized consumers this is from 2002 onwards and for the small consumers this will be possible from the year 2007 (or maybe earlier).

It is expected that this will result in lower prices for the consumer, which stimulate higher efficiency, new investments and cleaner production processes.
In the flexible, innovative telecom sector, the consumers have had profit by lowered prices due to competing providers for some time already. The same is going to happen in the energy sector of electricity and gas.

For the electrical energy production and distribution companies, this means that the production and distribution of their products has to be done in a more efficient way.

[1] In Dutch: 'Elektriciteitswet'
[2] In Dutch: 'Gaswet'
[3] KPN: Koninklijke PTT Nederland

One of the ways to achieve this, is by the extension of automation at the several larger and smaller energy distribution sites; the introduction of RM (Remote Monitoring) and data communication is one of the ways to come closer to his goal.

Chapter 3. Remote Monitoring

As seen in the previous chapter, Remote Monitoring is a way of achieving condition assessment for the use of Condition Based Maintenance. This chapter mentions the potential benefits of Remote Monitoring and Control, when applied to this system. The potential problems or points of attention are also discussed briefly. If applicable, possible solutions are mentioned, as well as a few recent developments that may eliminate problems.

3. Remote Monitoring and Control

This chapter explores the use of Remote Monitoring and Control of embedded systems via an Internet connection.
Several aspects described below are of interest for both Maintenance Management (MM) and Energy Management (EM).

Current situation
The current situation is that every HV-substation has its own Local Management Center (LMC), staffed by a local operating team. The equipment can be monitored and controlled by the LMC via the equipment's (embedded) systems.

The equipment within a substation produces a lot of information resulting from measurements. The information is collected and processed at the LMC, initially for the use of Energy Management.

Future situation
The extension to remote monitoring and control allows the management of substations to be located outside to the LMC.

A possible way of achieving remote monitoring and control is by the use of Internet. This chapter provides information on the benefits and possible technical difficulties, when using the Internet for Remote Monitoring and Control. However, there are several difficulties that should be dealt with.

3.1 Benefits

A few of the benefits of web-controllable systems are standardization, remote monitoring and control, software updating, and intelligent behavior.

3.1.1 Standardization

The use of Internet and its corresponding Internet Protocols stimulates **standardization**. The Internet Protocols (like TCP/IP, FTP and HTTP) have the benefit of years of real world testing on a variety of systems and networks.
An Internet connection enables the operator to interact with the MICS through standard browser software instead of built-in hardware switches and displays. This simplifies the new-product learning curve, because most users are already familiar with e.g. Microsoft Internet Explorer or Netscape Navigator. The operator will be able to control the remote system from any desktop computer. Beside that, it makes the system function independently from the platform (PC or UNIX-workstation).
Use of standard 'off-the-shelf' hardware lowers the system's costs and provides multiple product sources.

3.1.2 Remote Monitoring and Control

Embedding an Internet connection allows the operator to **remotely monitor and control** the equipment's functioning and condition, independent of place and time. The collected data may be used for several types of (distributed) applications, like statistics, quality control, remote diagnostics, backup and recovery, and planning of maintenance.
For example, traveling costs of maintenance engineers will be saved when remote diagnostics is being used. Planning of Maintenance can be based on Condition Assessment results, derived from both actual and historical data. A better timing of maintenance improves the QoS and also minimizes the probability of failures due to late maintenance.

3.1.3 Software Updating

The introduction of Internet also facilitates **Software Updating**. Bugs can be fixed immediately by downloading proper upgrading software at a suitable time. Downloading e.g. new control algorithms makes it possible to upgrade the system's functionality (e.g. the latest diagnostics software, or adding new features).

3.1.4 Intelligent Behavior

Intelligent Behavior is probably the most important benefit of a Web enabled system. Examples are remote documentation and autonomous behavior.

Because the configuration of a device or certain part of the equipment may not always be clear, the communication network may offer assistance via remote documentation. Intelligence allows direct linking of the operation of the system with the appropriate section of the documentation. A help function that has been called more often than others indicates that the function involved may be unclear.

An example of autonomous behavior is the generation of (asynchronous) system alerts in order to get the user's attention. Depending on the urgency, an email message may be used for alerts and warnings. The recipient can be anyone anywhere in the world, including someone back at the office of the device's manufacturer.

3.2 Difficulties

Next to all the benefits of embedding Internet into equipment, there are also some difficulties.

3.2.1 Reliability

Complex networks are not always reliable. Therefore, a primary design rule should be to ensure that the target system continues to function when the network

connection is down. Design of the system's hardware and software should lead to the use of the Internet to enhance the system's operation, not to be dependent on it.

3.2.2 Security

If the MICS's communication network is part of a network that unauthorized users can access, the system's security will be an issue. Some security can be got by using password-protected access or by restricting the system's response to unknown-user IP addresses. In some cases, however, the additional protection of a proprietary design is needed.

Designers also use firewall software and data encryption to foil hackers.
One of the newest security techniques is to set up a virtual private network (VPN), which allows secure communications through the public Internet.

Security may also be improved by having a security related layer close to the application layer, e.g. the emerging standard Secure Socket Layer (SSL) to protect application level data.
The SSL services include amongst others: authentication of client and server, and encryption of the application data traffic.

3.2.3 Feature Interaction and Simultaneous Usage

As the number of features in a system like this may grow rapidly, also the complexity of managing their interaction increases. It should be prevented that one feature conflicts with another. This complicates the development of new features.

The embedded system needs protection against simultaneous access by multiple users and the potential resulting inconsistent use.
This is solved by the concept of session: each session is dedicated to one (set of) user(s). During a session, certain commands are restricted to the participants of this session only; other commands are still available for other users.

3.2.4 Real-time constraints

Although the bandwidth demands of the system are relatively low, some information (e.g. system alerts) has to be transferred with a certain maximum allowed response latency. Recent developments allow new generation Internet protocols like IPv6 to assign priority to certain Internet Packets, (like e.g. real-time video, or to a MICS-system alert).

Embedded systems usually require an operating system (or Real-Time kernel) to handle interrupts and to provide context switching between tasks. With an Internet connection added, care must be taken that the extra Internet processing time should not cause a slower processing of the local control tasks (robustness of control loops). An example could be that the system gives higher priority to the processing of analysis results in case of a critical error, than to e.g. backing up log files on a remote system.

Because of its lack of determinism, the Internet should not be included inside any kind of control or feedback loop of the system. Instead, the Web should be used to collect and communicate data, whereas all the time-critical functions should remain under local control (locality of control loops).

The use of high-level languages for embedded systems software development tends to create larger and slower programs than would be produced by assembly-level programming. Also the execution of high level programs is less predictable with respect to memory usage and execution time.
An example is garbage collection, which is used to free unused parts of the data memory within programs. Its unpredictable behavior should be taken into account.

3.2.5 Resource Constraints

Resource constraints are for example size and costs of processors and memory.
On the one hand, the embedding of an Internet connection requires additional resources, like memory and processors.
On the other hand, resources like local memory can be reduced at the same time, because the Internet connection makes remote data storage possible. This aspect is especially interesting for large backup- and log files.
However, in this case the real-time behavior of the system will be more difficult to guarantee. Locally buffering these data and only sending them during times of sufficient processing power allows a trade-off between buffer capacity and real-time delay.

Chapter 4. Analysis and Specification Approach

Having seen the system's context, an approach must be chosen to develop the target system. This chapter discusses the approach of analysis, specification and modeling.

4. Analysis and Specification Approach

This chapter discusses the way of specifying complex Hardware/Software systems. Starting from an initial system's idea, the way to come to design and implementation of a certain system is often oversimplified. Analysis and specification lay a solid foundation of new systems. To avoid ambiguities, a well-formed specification of requirements and a specification of the system's behavior are of utmost importance to systems' designers.

Section 4.1 explains a possible approach to identify both requirements and limitations. It will also discuss the organization and representation of requirements.

A model lays down the system's behavior in a formal specification. The modeling technique and methodology that has been used will be explained in section 4.2.

4.1 Requirements development

Gradually, the importance of an unambiguous specification should have become clear, by now. This section describes the specifying approach. The capturing process will be discussed. An introduction to the various audiences is included.

The way of organizing and presenting the resulting document to the different audiences is at least as important as the way the actual requirements are deduced. Because the specification document will be used as a communication tool between several disciplines, it has to be clear to all. The people concerned should be able to read and understand the document. It also serves as a basis for further study or design and implementation decisions.

4.1.1 SyRS methodology

Special attention has been paid to the specifying methodology.
There will be many ways to achieve the creation of a document like this. However, the methodology that has been followed during this project will be highly recommended. The IEEE standard 1233-1996 ([53]) provides the analysts guidance for capturing system requirements, as well as a clear definition for identifying well-formed requirements and ways of organizing them.
A reason for using the IEEE document as a basis is that the methodology has been carefully thought-out by a well-established organization. A second reason is to familiarize people with this standard, nursing hopes for standardization of (future) specification documents within the company.

This project focuses on the benefits during systems design, leaving the other benefits aside.

The way of dealing with the dynamics is by identifying and communicating the requirements that are subject to (future) changes.

A requirements specification is a unique set of consistent requirements of a system within its boundaries. These three most important properties of requirements (with respect to this project) will be explained briefly, hereafter.

Uniqueness

Each requirement should be stated only once.
Only essential information should be in the requirements, while redundancy of information must be avoided.

Consistency

The System Requirements Specification contents should be consistent and non-contradictory in the level of detail, style of requirement statements, and in the presentation of material.
If the specification would contain items that can be interpreted in several ways, it will be evident that a system's designer and customers have to argue about the system afterwards. This, of course, will only consume (expensive) time and won't change the relation between customer and designer in a positive way.

Bounded

The boundaries, scope and context for the set of requirements should be identified. To eliminate endless discussions afterwards, it is very important to define the system's boundaries and keep them in mind. Unbounded requirements cause infinitely expanding documents that will never be finished. So will the system.

Other properties that requirements should have (less important for this project) can be found in [53].

The SyRS template that the requirements are based on can be found in appendix A. A combination of IEEE standards 1233-1996 and 830-1993 [54] have been used. The latter one ('IEEE Recommended Practice for Software Requirements Specifications (SRS)') contains items that do not appear in the SyRS standard, although they may me quite useful to the MICS specification.

4.1.2 Process Overview

Now that the choice of standard has become clear, this section summarizes the requirements development process overview.

Figure 4.1 shows the development process:

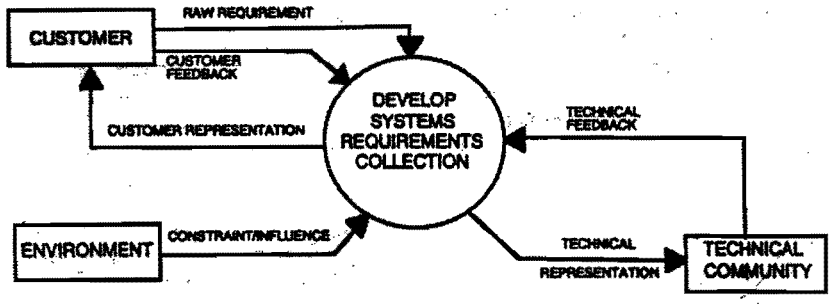


Figure 4.1 Context for developing an SyRS

The three external agents will be defined below:

Customer
The party that wants to develop a system will be referred to as 'customer'. The customer will draw up the target system's raw requirements, providing objectives, needs, or problems to the SyRS process.

Environment
The environment lays down constraints to the system or may influence some decisions. Examples are political, market or physical constraints.

Technical Community
The technical community consists of all sorts of analysts that are involved in the activities of system analysis, modeling and design, implementation experts, manufacturers, operations and maintenance.

The raw requirements that have been drawn up by the customer will be processed and presented to both customer and technical community in a standard format. Both of these two parties will give their feedback on the requirements collection. After several iteration, the System Requirements Specification may be used as a basis for further investigations.

The resulting specifications can be found in the next chapter (5).

4.2 Modeling Approach

The following will describe the approach that has been followed for the modeling of the system. First, section 4.2.1 will briefly mention some of the characteristics of OO (Object Oriented) Modeling Techniques, in general. Section 4.2.2. describes the SHE (Software/Hardware Engineering) methodology, on which the investigations in this project will be based.

4.2.1 Object Oriented Modeling

The use of an Object Oriented Modeling technique has several benefits, compared to conventional methodologies.

In the first place, OO-modeling stimulates better understanding of requirements. Graphical visualization of textual requirements is language-independent.
On the contrary to traditional data flow methodologies, an OO-methodology combines both data structure and behavior in a single entity. Cleaner designs are the result of this approach.
Apart from that, real-world concepts have more appeal for the people involved than computer concepts do.
Because the object instances used in the system are withdrawn from an object class, the system becomes more maintainable. Changes made to an object class are automatically carried through all object instances, which means that they do not have to be changed separately.

Analysis

The first phase of system development is analysis.
Object Oriented analysis extracts essential aspects of the application domain. It stresses the question what the system does and how. Objects are to be identified in the application domain, and their essential properties and behavior have to be defined.
During the analysis, no implementation decisions are made.

Design

The analysis phase is followed up by design. High level decisions are made on overall architecture, including subsystems. In this phase, decisions can be made on what performance issues are to be optimized.
A strategy to attack the problem is chosen in the design phase as well.

The design of object classes (low level) follows after high level design. Implementation details like data structures and processes of the object classes are included.

Implementation
In the last phase, finally, the designed system can be implemented.

4.2.2 Software/Hardware Engineering (SHE) Methodology

SHE is an OO-modeling methodology for system design. It aims at co-specification and co-design of complex hardware/software systems.

The process of system development from raw requirements towards system integration is an iterative one (Figure 4.2).

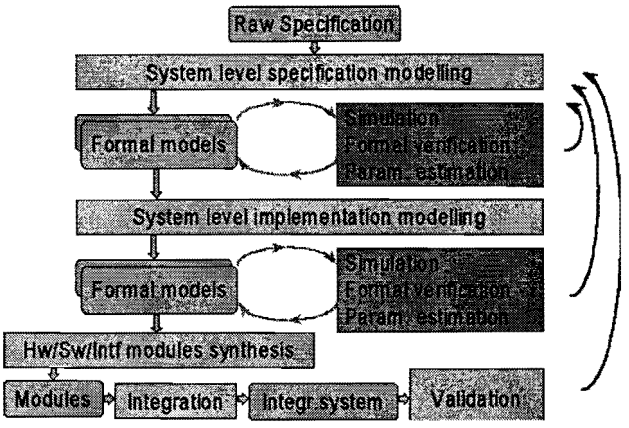


Figure 4.2 Co-specification and system level design

SHE captures system level specifications into a formal model with several views. The system's behavior can be specified in POOSL (Parallel Object Oriented Specification Language).
The simulator included (SHESim) allows simulation and verification of the specification.

In SHE, the process of specification and design are partially parallel. (Figure 4.3)

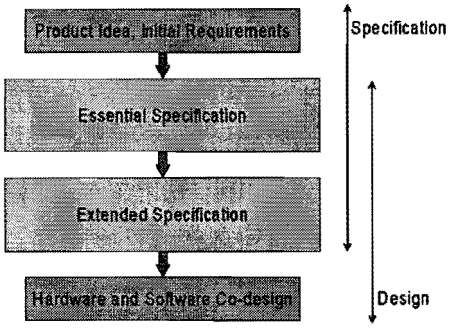


Figure 4.3 Process overview of specification and design

The process starts with a product idea or initial requirements. The SyRS document may be used as initial requirements.
An exploration of the problem domain results in an essential specification of the system's behavior, represented by several views.

Subsequently, details on implementation are included and several choices need to be made on several aspects, like architecture, and communication protocols.
The extended specification includes this additional information.

The design process starts with the choices that are made during the specification phase, in the essential and extended specification.

Eventually, hardware and software are to be co-designed, leading to the actual implementation.

As mentioned earlier, the formal model of SHE can be presented by several views, in the form of models. The individual models in Figure 4.4 will be explained briefly below.

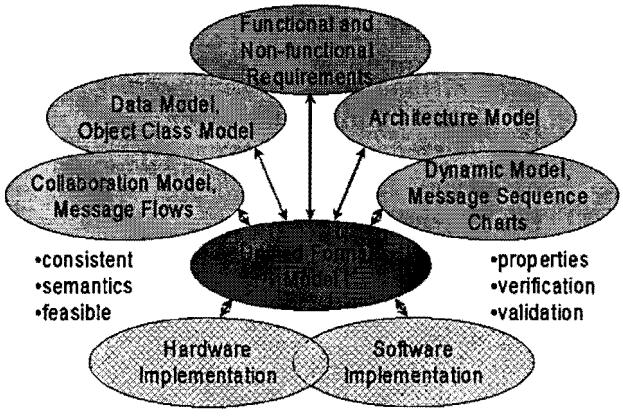


Figure 4.4 Representations of the formal model

Data Model

The data model is used for the analysis of the problem domain. The 'things' of the problem domain correspond to object classes in the model. The definition of classes belongs to the process of modeling.
SHE Objects can be characterized by their attributes and messages. For example, every transformer in the MICS may have a unique identity, which is an attribute of the class 'transformer'. A message that can be communicated to this object could be 'go_offline'.

Object Classes
In SHE, there are different types of classes: data classes, process classes and cluster classes.

data class:
A data class defines a certain type information, along with its typical operations. A data object is a dynamically created type of entity, which behaves in request.

process class:
Process classes represent the 'things' of the problem domain, mentioned earlier in this section. This means that it contains both information and has behavior. A process class is a type of entity that is interconnected statically and behaves autonomously. Messages are used for the communication with other process objects.

cluster class:
Although a cluster classes is a collection of process classes (and clusters), it does not necessarily have behavior of its own. The behavior is specified by the process classes inside the cluster.
Clusters allow us to use different levels of abstraction, as the internal structure of clusters can be hidden. The hierarchy of objects in the datamodel can be defined, which lays the foundation for the architecture model.

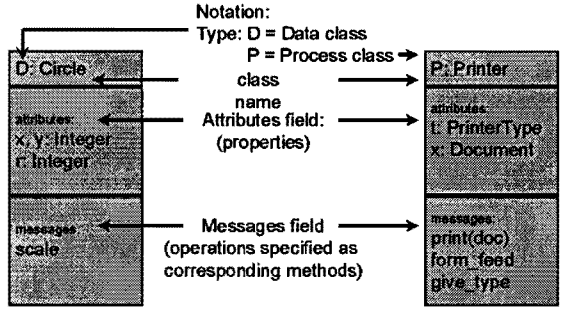


Figure 4.5 Class symbols

Relationships

The Object Class Diagram (OCD) visualized the relationships between object classes. The most important types of relations for this work are: aggregation and generalization.

Aggregation

Classes that consist of other ones, can be related by aggregates (Figure 4.6)

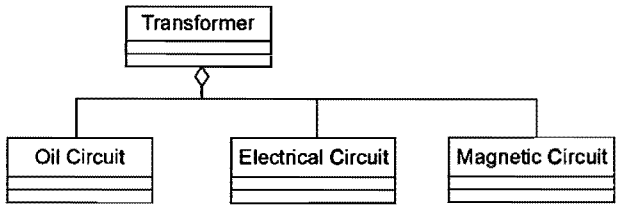


Figure 4.6 Aggregate relationships

The hiding of process classes in cluster classes can be recorded via aggregates, or 'part-of'-relationships. If a cluster has behavior of its own, the aggregate is a process class as well. An example is the relation between transformer and its oil circuit. The oil circuit is a part of the total transformer. Other parts of the transformer are denoted in the same way.

Generalization

Classification of objects can be identified by so-called 'is-a' relationships. This type of relationships is called 'specialization/generalization' (Figure 4.7).

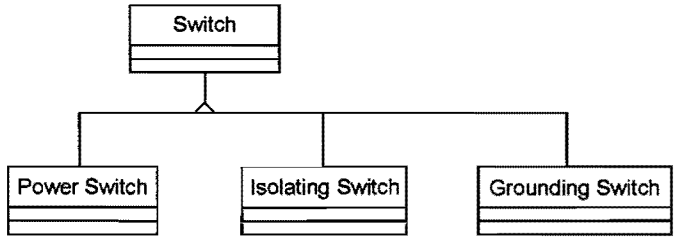


Figure 4.7 Generalization-specialization relationship

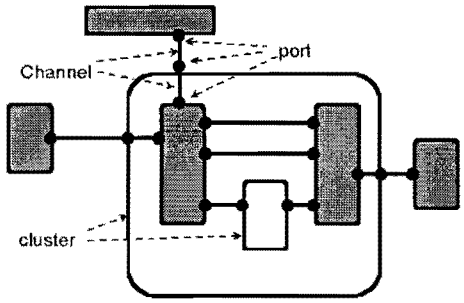
An implementation of the above mentioned analysis of 'is-a' relations, is the use of inheritance, which can be explained, by the example of switches: Many

types of switches can be identified, amongst others power switches, isolating switches and grounding switches.
A specialization (*child*) of inherits the properties of its superclass (*parent*), which means that the attributes and messages of 'switch' also apply to 'power switch', for example.

Architecture Model

The goal of architecture design is to define an adequate structure, visualized as a reusable collection of interconnected clusters and processes. Design considerations may be the physical topology, choosing either a centralized or a distributed system and the definition of boundaries.

The architecture model, which is called an 'Instance Structure Diagram' (ISD), consists of instances of classes and interconnections (Figure 4.8).
The ISDs may apply to various several levels of abstraction, e.g. system level and cluster classes level.



Cluster: a structure of process objects and clusters.
Cluster class: a (reusable) specification of this structure.

Figure 4.8 Instance Structure Diagram of a Cluster

An important aspect of architecture modeling is the realization of connections between process classes. The interconnections ('channels') connect to the 'ports' of the process classes' instances. For each connection a choice has to be made on the use of channels, whether to share a channel with several objects, or to use point-to-point connections. The mapping of the connections onto channels influences the behavior and messages of the process classes, vice versa.

Cluster classes are an example of reusable structures. The architecture within a cluster that interconnects the object instances inside, is part of the architecture model.

Some examples that apply to the MICS can be found in section 6.5 (simulation and verification).

So far, the static views of SHE have been explained. For the analysis of the behavior (dynamical views) other models are used.
A number of scenarios can be defined to distinguish between several types of behavior. Each scenario consists of a narrative (explaining the behavior in words), a collaboration model, and a dynamic model.

The models are explained below.

Collaboration Model

Process classes that are interconnected in the ISDs (Instance Structure Diagrams) communicate via messages. Message Flow Diagrams (MFDs) contain the messages from and to object instances, indicated by arrows. There are several types of message flow symbols for the MFD, but this work only uses one. Information on the other ones can be found in [34] and 0.

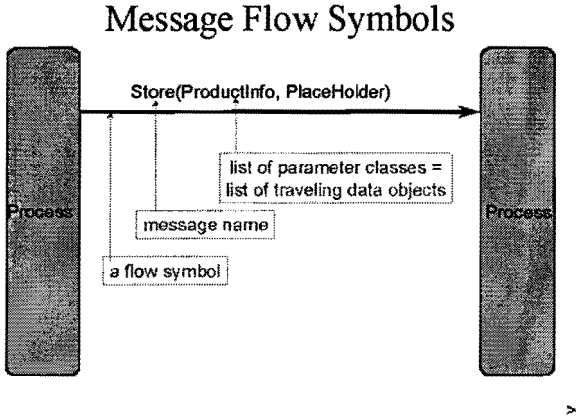


Figure 4.9 Message Format

A message from one object instance (process) to another has a message name and may contain a list of parameter classes (list of traveling data objects).

Examples that apply to the MICS can be found in section 6.4 (collaboration model).

Dynamic Model

The Message Sequence Charts (MSCs) show the dynamics of the message flow. For each process object, a vertical bar exists. The sequential order of messages is visualized downward in time.

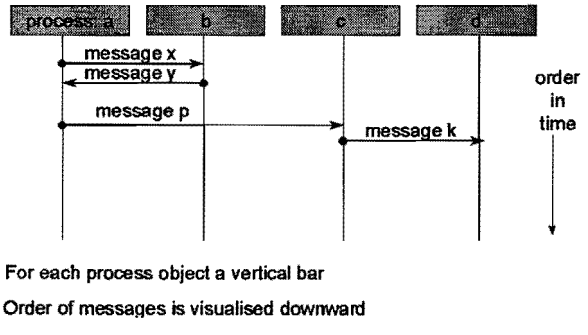


Figure 4.10 Message Sequence Charts covers time issues

Apart from the real-time aspects that can be identified, the MSCs will be of great importance to the visualization of parallel behavior in the system.

Some examples can be found in section 6.4.

Figure 4.11 visualizes / summarizes the different representations of the essential and extended specification...

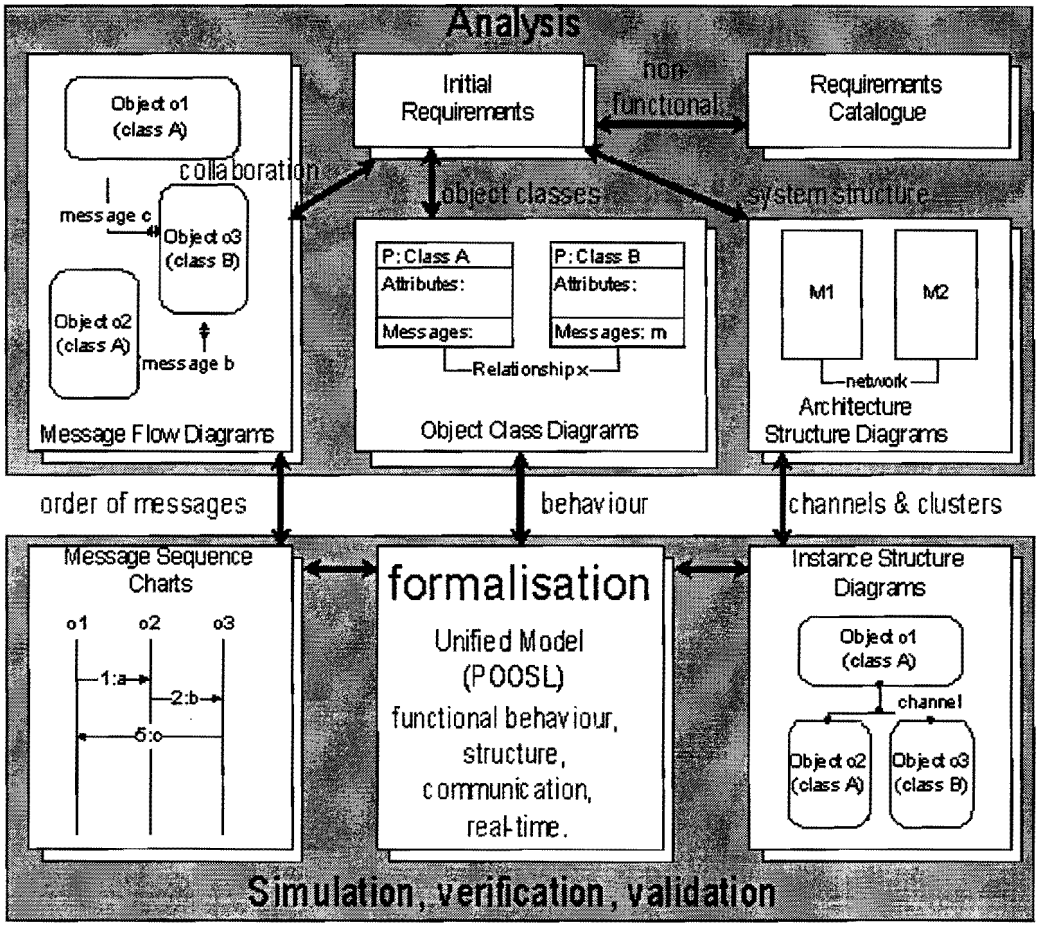


Figure 4.11 SHE method overview

The use of the simulator will be explained in chapter 6.

More information on the SHE methodology can be found at [46] and [47].

Chapter 5. System Requirement Specification (SyRS)

After discussing the approach of the analysis, specification and modeling, now is the time to formally specify what the system should do. The formal specification will be used in future investigations on the way from design towards implementation, as well as in seconding design and implementation decisions.

Due to the limited number of interviews, however, this chapter mainly contains limited KEMA input and some of the author's input. At the time of this study, not enough resources could be allocated to set up a real discussion group.
Elaboration of this chapter is required for future continuation of this project.

5. System Requirement Specification (SyRS)

5.1 Introduction

5.1.1 System Purpose

Purpose

The primary purpose of the system is to enhance maintenance of equipment in energy transport and distribution networks ('the electrical utilities') in the Netherlands.

The two main reasons are *cost reduction* and *quality improvement*.
Costs and quality are conflicting goals. This document will treat them separately. Eventually, it is the network manager's task to find a compromise between costs and quality.
Both goals can be achieved by improving and enhancing the maintenance system, with the aid of this Maintenance Information Communication System (MICS), facilitating Condition Based Maintenance (CBM).
The MICS will also be an aid in the *seconding of either strategic or operational decisions* on either cost reduction or quality improvement strategies.

To answer these purposes of the system, the individual goals will be worked out below.

Cost reduction

Cost reductions include both operational and non-operational maintenance costs.

Operational maintenance costs:
Costs on personnel and equipment/material can be reduced by introducing Remote Monitoring and by improved understanding of the internal equipment's conditions with respect to foreseeing failures.

The benefit of Remote Monitoring is that the presence of personnel at substation sites will no longer be necessary. It also allows centralization of the experts team. Costs of man-hours will go down due to less traveling time for inspections of substations.
The more autonomously the network function, the less (local) user intervention is necessary.
As failures can be foreseen in an early stage, necessary maintenance can be scheduled earlier in time. The availability of replacement parts (regarding delivery time) will be higher. A historical database on system level with enhanced features like e.g. detection of transformer type specific failures, may be an aid in the improvement of the knowledge of equipment.

Non-operational maintenance costs:
The costs of investments will be lowered, due to the extended life span of equipment. The increased utilization can be achieved by more accurate information on the actual need for maintenance.
The liability costs, resulting from non-deliverable energy, will decrease with the increase of the network's availability.

Quality improvement

Specifically targeted maintenance, provided by better information supply, improves the quality of maintenance. The knowledge on specific network equipment will also be extended by the increased availability of information.

The improvement of service depends on the reliability of power delivery. Less and shorter outages promises better service.

Intended audience

The intended audience of this SyRS (System Requirement Specification) is the Dutch utilities, as well as the federation of energy related companies in the Netherlands, EnergieNed.

Furthermore, this SyRS will also be relevant to KEMA, in their role of independent advisory and cooperator.

Finally, the Eindhoven University of Technology (EUT) belongs to the intended audience as well. It is highly probable that they are willing to assist the process of design, considering the available knowledge on the followed modeling methodology.

5.1.2 System Scope

Identification of the system

The system to be referred to as the 'Maintenance Information Communication System' (MICS): the system communicates maintenance information.

Results of earlier analysis

As the results of earlier analysis, the following reports have been identified:

'INSITE Continuous Diagnostic Information System' [8]

Summary:

> *INSITE is a comprehensive, online, substation diagnostic information system, which uses continuous measurements to detect apparatus malfunctions in the incipient stage.*

'Glass: Remote Monitoring of Embedded Systems in Power Engineering' [7]

Summary:

Glass (Global Access for Service and Support) is an example of a monitoring system to control and manage industrial embedded systems. It demonstrates the benefits of Internet in power engineering.

Conclusion:

Inventive way of extending the life span of existing embedded systems and of increasing productivity and efficiency of maintenance teams.

'Raamwerkplan Informatie en Communicatie Technologie voor energiebedrijven' [3]

Summary:

This report discusses the results of the liberalization of the energy market, concerning information and communication systems.
Liberalization separates energy companies into 'production' and 'transport', two individually operating divisions (or even two different companies). The 'production' part are occupied by the trading of energy, while (operational) management and network development concerns the 'transport' part (network companies).
Regarding this requirements specification, the latter is the more interesting part, which can be subdivided into 'protection and control' (1), 'network management' (2), and 'network development and planning' (3).
Maintenance management, which is the central point of this specification, can be found in the 'network management' section. The report [3] describes different levels of information management.

Conclusion:

The main conclusion is that relating different information systems is a necessary condition to survive for energy companies.

As 'foresight is the essence of government', the added value of integrating the MICS into the existing collection of collaborating systems appears from the expectation for the future. Noticing a trend towards fully autonomously functioning systems, without human intervention, emphasizes the need to study these matters and possibilities in advance.

System application description

The main reason for designing a MICS is the ability to reduce maintenance costs by applying the strategy CBM to equipment in electricity networks. The system should be able to communicate information on the equipment's conditions from a certain source to a predetermined destination.

Chapter 3 has already dealt with 'Remote Monitoring (RM)' as a technique to capture and communicate information remotely, e.g. over the Internet.

5.1.3 Definitions, Acronyms and Abbreviations

Onsite	Locally
Remote	Not located at the substation
Online	In operation
Offline	Out of operation
SAS	Substation Automation System
MICS	Maintenance Information Communication System
END	Energy Not Delivered
HV	High Voltage
MMI	Man-Machine Interface
HMI	Human- Machine Interface
MV	Medium Voltage
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SyRS	System Requirement Specification
TUE	Eindhoven University of Technology (*Technische Universiteit Eindhoven*)
EMS	Energy Management System
TDS	Transformer Diagnostics System
IED	Intelligent Electronic Device
QoS	Quality of Service
NM	Network Manager
ROI	Return on Investment
BOM	Bill of Materials

5.1.4 References

Complete list of all documents

This document shall be used in conjunction with the following publications:

[1] (Standard)
IEEE GUIDE FOR DEVELOPING SYSTEM REQUIREMENTS SPECIFICATIONS (STANDARD 1233-1996)
New York: IEEE, 1996.

[2] (Standard)
IEEE RECOMMENDED PRACTICE FOR SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) (STANDARD 830-1993)
New York: IEEE, 1994.

[3] Dijk, H.E. and F.L. de Rooy
RAAMWERKPLAN INFORMATIE EN COMMUNICATIE TECHNOLOGIE VOOR ENERGIEBEDRIJVEN – CONCEPT.
Arnhem: KEMA Nederland B.V., dept. of KEMA Connect, 1998.
Ordering Information: Report 5007.CON18.P01 (98340007-Connect 98-xxxx).

[4] Lehtonen, M.
FAULT MANAGEMENT IN ELECTRICAL DISTRIBUTION SYSTEMS

Espoo: CIRED, Working Group WG03 Fault Management, 22 December 1998.

[5] Antal, M.
BEHEERSASPECTEN VAN DE ENERGIEVOORZIENING EN TELECOMMUNICATIENETTEN.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, August 1992.
EUT Report 5756.

[6] DeZwart, T and J.K. Stoutjesdijk, Y. Fu, H.F.A. Verhaart, G.P. Krikke, C. Kroon, J.W. VanderStoep, W.R Rutgers, F.B.J. Koops
CENTRAM TRANSFORMATORDAG.
Arnhem: KEMA Nederland B.V., 1999.
Contact Information: Centram, P.O. Box 9035, NL-6800 ET, Arnhem, Informant: W. DeLeeuw, Tel.: +31 (0) 26 356 23 23, Fax.: +31 (0) 26 351 36 83, Email: centram@kema.nl.

[7] 'Glass: Remote Monitoring of Embedded Systems in Power Engineering' http://computer.org/internet; IEEE Internet Computing May/June 1998

[8] 'INSITE Continuous Diagnostic Information System' http://www.doble.com/doble/marketing/insite.html; Doble

5.2 General System Description

The MICS aims at 110 and 150 kV (HV) transportation networks. The number of HV-substations can roughly be estimated at 200. The costs of purchasing and maintaining the MICS will be relatively low compared to the high costs of HV-substations.
In contrast, the number of MV-substations is relatively high. In general, the savings will count for little, compared to the investments of system integration.

Nevertheless, when the system appears to be very successful (and prices drop), companies can always decide to integrate the lower voltage networks as well.

The first step is to explore the system's environment and its role in a complete electricity network. External interfaces on the interaction between different systems across the MICS's boundaries have to be identified and defined.

Next to the behavior of the system as a whole in its environment, the internal behavior of the system should be clarified as well. Both parts will be included into the specification. In the future, implementation decisions can be made on the basis of this document.

The central questions 'what information' should be monitored 'when' and 'by whom', will be answered briefly in this section.

5.2.1 System Context

Figure 5.1 gives an impression on the MICS (inside the cloud) and the interaction with its environment. The external systems, outside the cloud's border will be mentioned briefly, along with their individual functions and their relations to the MICS.

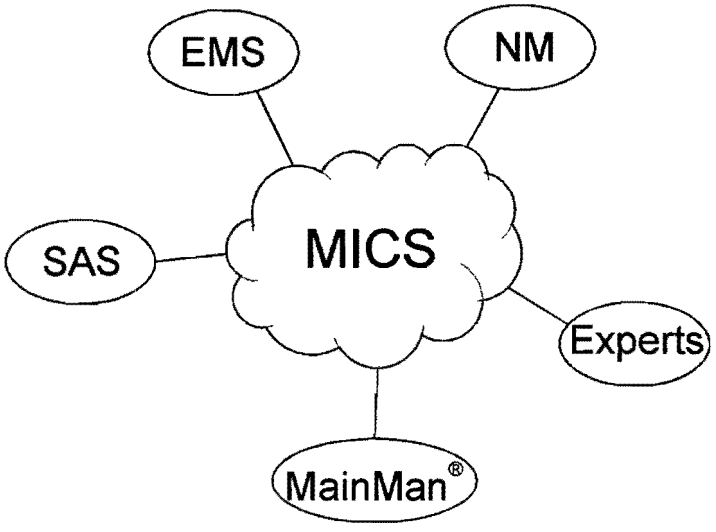


Figure 5.1 MICS's system environment

In the first place, the Network Manager (NM) is in control of the MICS.
A close interaction with the Energy Management System (EMS) will be necessary to relate the aging of components to the capability of carrying a certain (over)load.
The Substation Automation System (SAS) that is available in most of the advanced HV-substations may provide easy access to the information (data) that is already available inside the substation. However, many separately developed systems exist, which makes standardization quite difficult. A dedicated interface to a particular SAS may be necessary.
Experts need to access the MICS when they need certain maintenance information.
MainMan®, finally, is an existing Maintenance scheduler that requires information from the MICS.

The interfaces concerned are to be defined later on.

With respect to the system's internal behavior, several alternatives exist, like the two below:

- communicating all information of all devices (components) as soon as available to all external systems;

contrary to:

- keeping as much information inside the involved (sub)system as possible, while communicating merely highly necessary information to external systems.

Intelligent alternatives may be chosen to keep unnecessary communications down to a minimum. This will save complexity of the system (and thus investments), while retaining both efficiency and effectiveness, at the same time.

All failures, alerts, alarms, automatic decisions, will be logged both locally and centrally. In Case of an alarm, the substations will handle autonomously by taking actions themselves or communicating an emergency signal to a central point, depending on the gravity of the situation.
Non-urgent measurement data will be kept locally within the substation. Updates to the central database will occur once a month, or on request.

5.2.2 Major System Capabilities

Condition assessment is the central point of the internal behavior of the MICS. Several levels of collecting information can be declared, from measurements by sensors to extensive diagnosis reports including e.g. life expectancy.

Currently, a number of elementary measuring sensors have been integrated into the equipment, so some quantities can be monitored 'on-site', i.e. on the spot. Some of these measurements can merely be done when the equipment is out of operation (called 'offline'). However, there are developments towards 'online monitoring', i.e. while the equipment stays in operation.
Most of the times, dedicated companies do the more sophisticated measurements and analyses, using very expensive and delicate equipment. A team of specialists visits the substation and measures all necessary quantities on-site. Back at its office, the team analyses the measurement results and makes a diagnosis. A report advises the customer (energy company) what action should be taken next or just communicates the measurement results.

In ten years, lots of measurements will be done online. A logical consequence is the development towards remote online monitoring. This eliminates the necessity for specialists to visit a location that should be monitored. The benefits are evident.

Another change to be expected in the near future, is the shift of autonomy. At this moment, a few people of the energy companies take the initiative to certain actions. Future equipment will be able to take its own initiative to execute some actions, or to report that intervention is necessary. In other words, the process of measuring, analyzing, diagnosing and decision making is being automated. The maintenance process will operate more independently, which makes the process more efficiently.

The internal part of the MICS is the part that collects information via sensors. The information is being analyzed and compared to historical data. The person responsible combines diagnoses with other data (like economical figures), and will follow the most favorable strategy.
Automating maintenance management is the next step in enhancing maintenance management.

The MICS should be able to share information (measurement data, analysis results, etc.) with other systems across the MICS's boundary (see Figure 5.1). Redundancy of information will be avoided.

Substations should be able to function autonomously. Examples are:
- signaling in case of abnormalities,
- alarming in case of critical failures;
- autonomously behavior in case intervention is required immediately,

The MICS is also capable of processing the assessed maintenance information, such as
- calculating expected remaining life time;
- scheduling next condition assessment;
- scheduling maintenance

The MICS should be able to build up a history of failures on specific type of equipment. If necessary, the NM can communicate this information to the manufacturer.

5.2.3 Major System Conditions

Beside the quality and reliability of the system, also financial aspects play an important role.
Because of the age and life span of certain machinery, a transitional period of at least 10 years will be required. Furthermore, 90% of the existing equipment should be supported by the MICS, otherwise the investment would be so enormous that no customer would be willing to participate.

Furthermore, the ROI should be less than 10 years.

The total BOM of the MICS plus integration costs have to be less than 10% of the current value of the substation.
The MICS should not cause a degradation of quality.

In case of a system failure, the MICS's autonomous behavior should be overrulable by manual control.

5.2.4 Major System Constraints

Regulatory policies

The MICS is not allowed to cause any failure of the power delivery.

Hardware limitations (e.g. timing)

to be defined

Interfaces to other applications

to be defined

Parallel operation

There may be multiple sessions of remote information monitoring, but multiple remote control sessions are not allowed.

Signal handshake protocols

Both industry standards IEC870-5 and TCP/IP should be followed.

Reliability requirements

The MICS has to be at least as reliable as the conventional system.

Chapter 6. MICS Model

After having discussed the modeling approach in chapter 4, and after the requirements of chapter 5, the time has come to demonstrate the modeling methodology 'SHE', using a basic and illustrative example model. Several views will be shown.

6. MICS Model

This chapter shows an example of a system, modeled in POOSL, using SHE. The example will show the process of specifying a hardware/software system, in this case a Maintenance Information Communication System (MICS), using an illustrative example.

During consultations with my coaches, a basic model has been chosen that demonstrates interesting aspects of the modeling process and the power of SHE. The analysis of the problem follows in section 6.1. Section 6.1.1 describes the basic model in general; the problem statement (6.1.2) more specifically describes the system of which the behavior will be specified in an executable model.

The sections 6.2 and further discuss the different views of the model. Next to the description of the executable model, several other aspects are brought up, that should (or 'could') be thought of for future investigations.

Section 6.2 mentions the derivation and description of the different process classes (6.2.1 and 6.2.2, respectively) and the description of the clusters classes (6.2.3). The internal relationships between all classes is shown in the Object Class Diagram of 6.2.4.

The architecture model in section 6.3 describes communications topology, and the mapping of messages of the MFDs (see 6.4) onto channels.

The collaboration model of section 6.4 describes the communication between different object instances, using Message Flow Diagrams (MFDs) and Message Sequence Charts (MSCs). A few scenarios are defined and explained. They illustrate the system's deviant behavior in a number of situations.

Section 6.5 discusses the simulation environment and shows some simulation results by the use of several screenshots.

Finally, section 6.6 gives an impression on experiencing SHE by describing the development process (6.6.1), some of the encountered problems and their solutions (6.6.2) and recommendations for further modeling of this system (6.6.3).

6.1 Analysis of the problem

This section describes the system to be modeled.
First of all, the problem should be analyzed. The requirements capturing process can be on the basis of the analysis, resulting in a system description (6.1.1) and a problem statement (6.1.2).

6.1.1 System Description

Every energy distribution network has a Network Management Center (NMC) that keeps control of the total electrical power transport and distribution process. This is the highest level of network management. The Central Energy Management System

(CEMS) belongs to the top-level of the management, and therefore it is part of the NMC.

The Network Manager (NM) is assumed to follow a CBM strategy (see 2.4.3). Several techniques can be used to determine the equipment's condition. Generally, they are all alike in a way, that they process measurement data and translate these measurements into information needed by the maintenance manager, for instance, the expected remaining lifetime, in order to schedule maintenance needs.

One of those tools is the Frequency Response Analyzer (FRA). The result of this offline analyzer gives the NM an impression on the mechanical condition of transformers. (These characteristics often change after displacements, movements, transport, or after serious shortcut-currents.)

Figure 6.1 shows the network, owned by a distribution company.

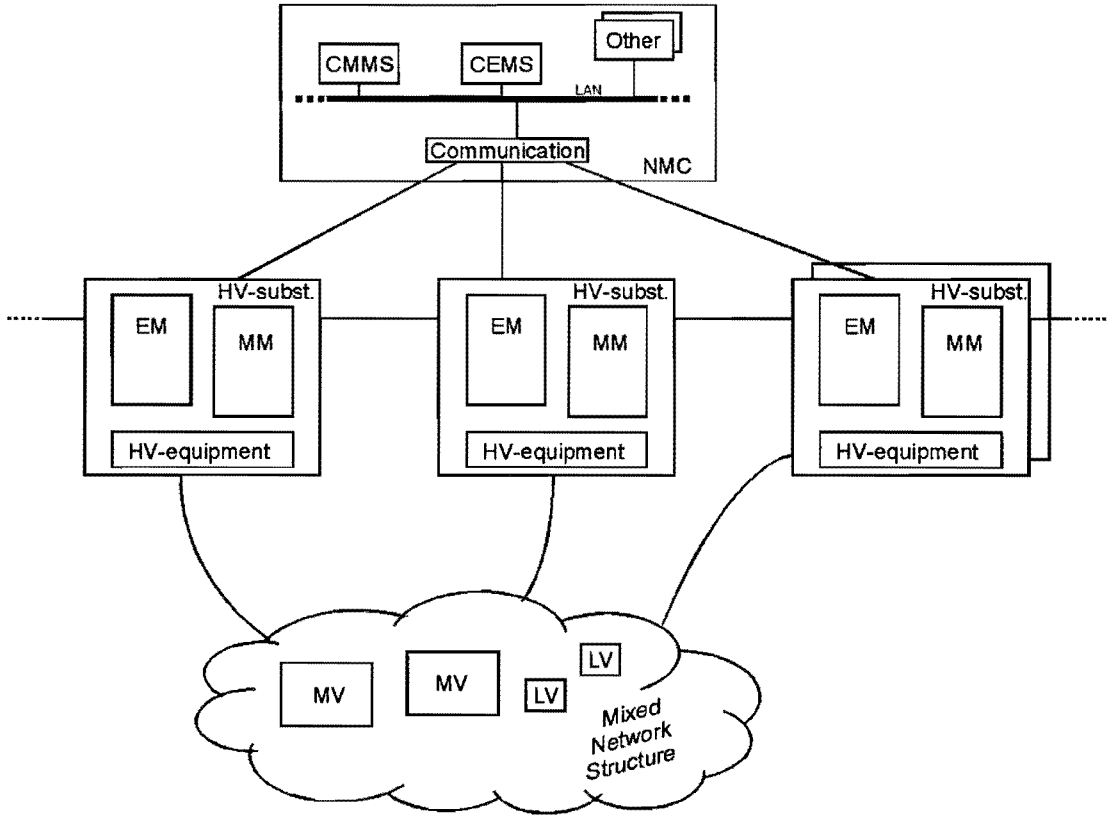


Figure 6.1 Network of a distribution company

The Network Management Center (NMC) on the top contains different systems. The two most important are the Central Maintenance Management System (CMMS) and the Central Energy Management System (CEMS). A communication system handles the transport of information from and to the NMC. Beside that, there may be other systems present that are of less relevance to the example model in this report.

The CEMS takes care of tens of HV-substation (EHV-substations are managed by TenneT). The substations can be seen as a layer below the system level layer containing CEMS. Each substation contains a number of transformers and other equipment, which can be seen as a 'third' level. The equipment itself is indicated by

'HV-equipment', while 'EM' (Energy Management) and 'MM' (Maintenance Management) indicate the corresponding systems inside the substations.

The NMC could probably have some kind of Local Area Network (LAN), which connects all the internal systems. However, the implementation is not important to this model; the ability of the systems to communicate with each other is.

There are several data-connections to HV-substations, using a star topology. This means that every HV-substation uses a point-to-point connection to communicate with the NMC.
For protection purposes, HV-substations also have a direct connection to adjacent HV-substations that follows the energy topology.

The cloud below is a collection of MV-transport networks, down to LV-distribution networks, with mixed structures.

The model focuses on the direct communication between NMC and HV-substations, although the NMC may have the ability to communicate directly to important MV-substations as well (not indicated in Figure 6.1).

6.1.2 Problem Statement

The Transformer Maintenance Agents (TMAs) keep a schedule, when they need a new analysis to be done by the Frequency Response Analyser (FRA). If so, the TMA initiates a FRA-process within the substation. Assuming that there is a physical device that performs the FRA, the device is being 'shared'. Before even starting measurements, the FRA should be assigned to the first requesting transformer. Every second request during the analysis process will be scheduled and immediate after finishing the first one, the analysis will be started for the corresponding transformer. Every other (third and on) request is rejected, until the current analysis has finished and a reservation place is available.

The transformer needs to be offline for the examination. The offline requests will be aimed at the Central Energy Management System (CEMS), whose permission is necessary to start measuring. This is done via the Transformer Energy Agent (TEA). A transformer in only one substation is allowed to be offline at the time.
For safety reasons, a cooling down period may be required, until the oil temperature is below a predetermined safe level. A dedicated Enhanced Intelligent Sensor (EIS) triggers TMA when ready.

After calculating the results, the transformer can be switched online again with the CEMS's permission.

In the next sections, the behavior of different objects concerned is shown in case of performing an FRA.

6.2 Object Class Modeling

The data model, or 'Object Class Model', is an aid in the exploration of the application domain. The object classes, which represent the 'thing' from the problem

domain, are specified in the form of attributes and messages. Later on, their behavior is recorded in a POOSL description.
First of all (6.2.1), the different process classes are derived from the problem description of the previous section. A description of each process class follows in section 6.2.2, after which 6.2.3 briefly mentions the cluster classes within the example model. The Object Class Diagram, finally, gives the classes' mutual relationships (6.2.4).

6.2.1 Deriving Classes and Objects

Nouns in the problem description can be seen as the 'things' in the problem domain. Not all of the nouns will eventually be defined as a separate class: discussions should lead to a selection.

The underlined nouns of the problem statement result in the following list of candidate classes: *Transformer Maintenance Agent (TMA), Frequency Response Analyzer (FRA), transformer, Central Energy Management Center (CEMS), Transformer Energy Agent (TEA), substation, oil temperature, Enhanced Intelligent Sensor (EIS).* In addition, the list can be supplemented with the nouns in the system description: *distribution network, Network Management Center (NMC), Network Manager (NM), Analyzer, and Central Maintenance Management System (CMMS).* All of these classes are useful in a model of the MICS. In the example shown in this chapter, only a selected number are used. The other ones give an impression on what may be included later on.

Section 6.2.2 describes each process class and its function; section 6.2.3 describes the cluster classes (reusable clusters of objects). Section 6.2.4, finally, gives an overview of the relations between classes, in an Object Class Diagram (OCD).

6.2.2 Definition of the Process Classes

This section describes the different process classes and their behavior in the example model.

CEMS (Central Energy Management Agent)

The Central Energy Management Agent has the task of handling offline and online requests. Its view on the total network allows it to make a decision on the basis of the current state of the total network.
In response to an offline request, the CEMS checks whether all transformers are online at that moment. If not[1], the CEMS gives the permission, meaning the transformer that initiated the request has to wait for a certain period of time and repeat his request.

Beside the identity of the request's originator, several parameters can be included in the request message sent by a transformer, like 'expected offline duration' and some kind of urgency (which the CEMS interprets as a priority. If, for instance, offline

[1] Here, we recall to the fact that merely one transformer can be offline at the time

permission is required for the purpose of maintenance, its urgency may not be as high as a critical error in another substation. The CEMS must handle an urgent failure situation with higher priority than the maintenance situation.
When a transformer is offline for maintenance purposes (after permission) while a highly urgent request is being received from another transformer, the CEMS may decide to abort the maintenance job, switch the component back online again, and give permission to the more urgent request.
Depending on the urgency, the CEMS may also decide to schedule a higher priority task directly after that the offline component goes back online again. The second component waits for the current offline component to go online again, and then goes offline immediately.

In these decisions, the estimated offline duration may also be of importance.

FRA (Frequency Response Analyzer)

Each substation contains only one analyzer of each kind.
The current state of availability can be 'busy', reserved', and/or 'under_maintenance'.
When the analyzer is under maintenance, it will not be available to any component.

If assigned to a component, the analyzer's current state changes into 'busy'. This state means the analyzer is measuring or processing data, and is therefore not directly available to other components.
When a second request arrives at an FRA in 'busy'-state, the FRA will also pass off to be available, but does not immediately start the analysis. A reservation is made, and the analysis will be started after finishing the current one.
The third request (and on) is being rejected, indicating that the analyzer is not available.

TMA (Transformer_Maintenance_Agent)

The TMA (Transformer Maintenance Agent) is a part of the transformer that schedules maintenance jobs.
Condition assessments are initiated by the TMA. Maintenance decisions will be made on the basis of the resulting transformer's condition.

When an analyzer (the FRA in this case) is asked to carry out an analysis, several conditions may be stipulated, that should be met before measurements can be started. The FRA requires the component to be offline with an oil temperature below a safe level. Offline requests are put out to the Transformer Energy Agent (TEA); temperature requirements should be checked by the process class T_oil. If the demands are met, the TMA reports so to the FRA. The FRA, subsequently, starts the process of measuring, analysis and diagnostics. When ready, the results are reported to the TMA again.

If the FRA is not available, the TMA schedules another request on a short notice.

If the conditions are met or one of them failed, it will be reported to the FRA.

For the use of global calculations on various populations of equipment (network wide), the TMA communicates with the Central Maintenance Management System (CMMS) regarding measurement or analysis results.

This example model does not make use of the CMMS yet, but later on, this class may be included.

Possible timeouts of communication with the FRA are handled by the TMA as well.

TEA (Transformer_Energy_Agent)

The Transformer Energy Agent (TEA) handles offline request and online requests from the TMA. Necessary permission is obtained from the CEMS. If permitted, the TEA executes the actual switching off respectively on, and confirms the action immediately afterwards.

When a critical situation occurs, the CEMS may decide to force a component (transformer) to go back online during maintenance. Also this is done via the TEA.

T_oil (Temperature_oil)

One of the sensors that are part of a transformer is the temperature sensor. The process class T_oil uses an image that represents the real oil temperature.
This example solely uses one of the sensor's capabilities:
T_oil can be asked to monitor the oil temperature, until it is below a certain value. This value is a parameter of the message sent to T_oil. If the temperature does not drop to that value within a predetermined time, measurements are interrupted and T_oil reports a temperature failure.

Other functions that are not implemented in this example could be:
- reporting when the temperature exceeds one or more thresholds
- second of strategic decisions in determining the extend in which a component can be overloaded for a certain period of time.

6.2.3 Definition of the Cluster Classes

Substation

A reusable collection of instances 'transformer' (multiple) and 'FRA' (single).
The instantiation parameters are 'substation_ID'.

Transformer
A reusable collection of instances: 'TEA', 'TMA' and 'T_oil'.
The instantiation parameters are 'component_ID' and 'substation_ID'.

6.2.4 Object Class Diagram - Relations between Classes

The relations that have been found between the different classes of section 6.2.1 are shown in the Object Class Diagram (OCD) of Figure 6.2. Two types of relations exist (and are present):
- aggregation (whole-part relations), and

- specialization (generalization-specialization tree)

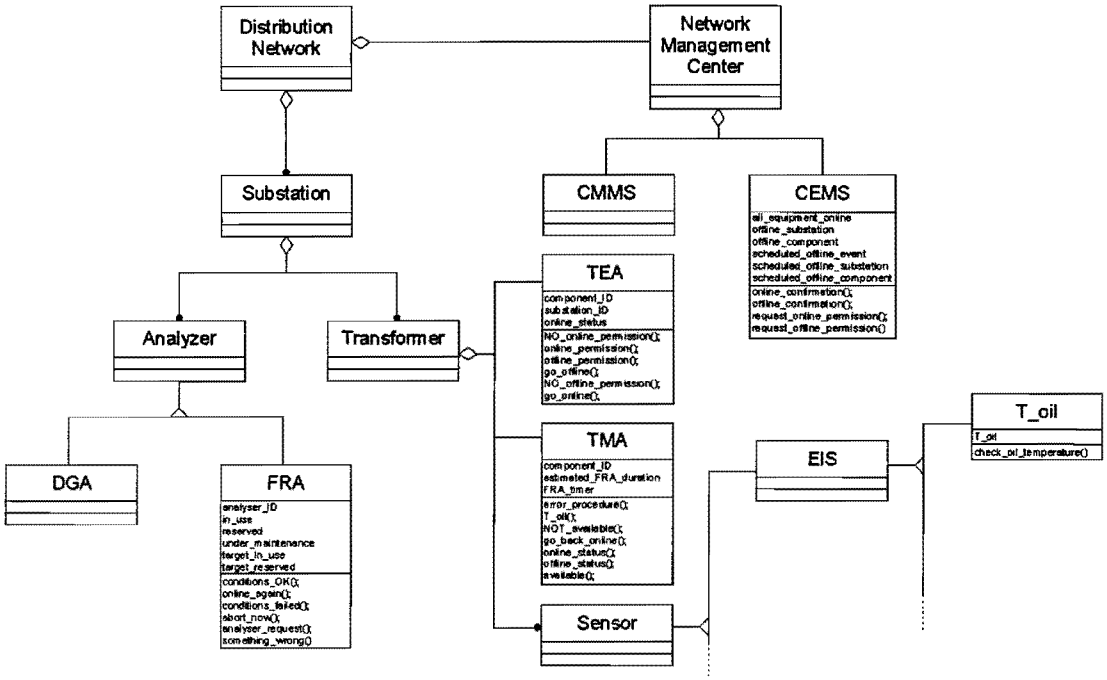


Figure 6.2 Object Class Diagram of the Process Classes

An aggregation relation (indicated by a diamond-symbol) can be found for example between 'Transformer', on the one hand, and on the other hand 'TEA', 'TMA', and 'Sensor'. It means that classes 'TEA', 'TMA' and 'Sensor' are **part of** the class 'Transformer'.

A specialization relation (triangle) can be found between 'Analyzer' and 'FRA', for instance, meaning 'FRA' is a *specialization* of 'Analyzer'. 'FRA' inherits all the attributes and messages of 'Analyzer' (as DGA does as well). The FRA also has a few typical attributes and messages of its own that cannot be found in DGA.

More information on the relations between classes can be found in [34]

As mentioned earlier in section 4.2.2, attributes represent the properties of an object class. An example may be the 'identity' of a TEA or the 'online-status' (whether a transformer is either online or offline). The 'in_use' attribute of process class 'FRA' indicates the direct availability of this analyzer (if in use, the FRA is not directly available, although a second request may be scheduled). The attribute 'reserved' indicates whether there has already been a second request. The sender waits for the availability of the FRA, which will be confirmed directly after the first analysis has finished.
The object instances interact via the exchange of messages. The behavior the process classes determine the messages to be sent or received over a certain channel.
The behavior of a process class in SHE can be specified using POOSL (sort of a programming language). The behavior description of each process class is captured in a number of 'methods'. Each of these methods lays down a piece of the behavior, which makes the specification more transparent.

Messages, though, are not necessary bounded to a certain method, vice versa. The methods allow structuring the total behavior of a process class, while messages are based on the required total behavior, not on its internal structure.

Eventually, the OCD can be extended with all object classes that are found later on in the continuation project. The discussion on whether some kind of behavior belongs to one class or its super class (generalization and inheritance) may change the contents of the classes. But the advantage of moving attributes or messages to a super class is that many attributes or messages can be reused, and the corresponding behavior does not need to be duplicated.

The results of an early exploration of the class 'sensor' can be found in appendix B

6.3 Architecture Model and Design

The model at system level contains one communication channel: between CEMS and the substations (Figure 6.3). This means that one communication channel is shared by multiple substations.
The channel between the object 'CEMS' (upper) and multiple objects of 'HV_substation' (lower) is called 'energy_management' and carries messages 'msg of energy_management'.

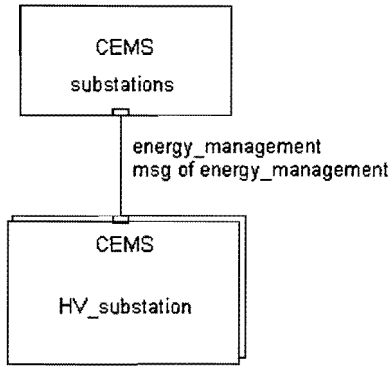


Figure 6.3 Instance Structure Diagram (ISD) at system level

The 'ports' of objects provide the connection to a channel and their names have been chosen corresponding to the object on the other side of the channel. This makes the messages more transparent.
In POOSL, the communication of a message from an HV-substation to the CEMS can be specified in the following way:
`CEMS ! message_name (parameter1, parameter2)` for the HV-substation, and
`substations ? message_name (parameter1, parameter2)` for the CEMS object.

Eventually, the point-to-point connections between substation and CEMS should be modeled as such; the example model does not foresee that yet. Only the behavior of the most important part of the NMC (see Figure 6.1) has been specified. The extension of the NMC with a communication object, via which the CEMS may communicate, remains an afterthought. The step towards the basic structure of Figure 6.1 has to be taken later.

The reason why the object CMMS has not been implemented yet, as may be expected, is that most of the attention has been paid to the behavior on substation and transformer level. Maintenance behavior on system level (e.g. updating the analyzers' diagnostics tool, or relating the analysis results of the various transformers within a certain product 'population') may be an advisable extension.

The Instance Structure Diagram (ISD) of Figure 6.4 shows all objects of the model and their 'physical' connections.
As can be seen, multiple transformers share the channel of energy management of Figure 6.3 within the substation.

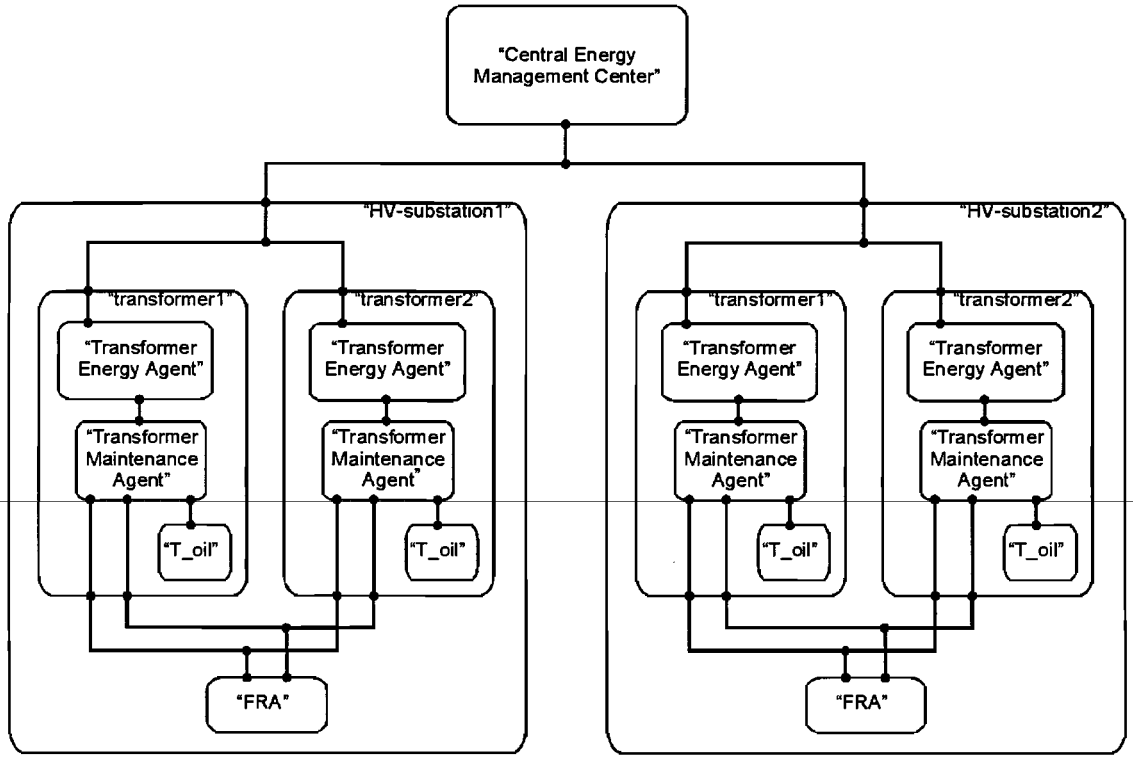


Figure 6.4 ISD of the total modeled system

The analyzer (FRA) uses two shared channels within the substation, to communicate with the several Transformer Maintenance Agents (TMAs): one for regular traffic, one for urgent error messages. The latter channel has to comply with more stern demands than the first one. Error messages may be sent, when the 'regular' channel appears to function incorrectly.
As can be seen within each transformer, the Transformer Maintenance Agent plays a central role in modeled scenarios. An example of an alternative is that the analyzer communicates directly with TEA (e.g. in case the analyzer needs the transformer to be offline).

Figure 6.5 is the ISD of the cluster class 'substation'. Here, also the names of both the ports of the object instances and the connecting channels are included.

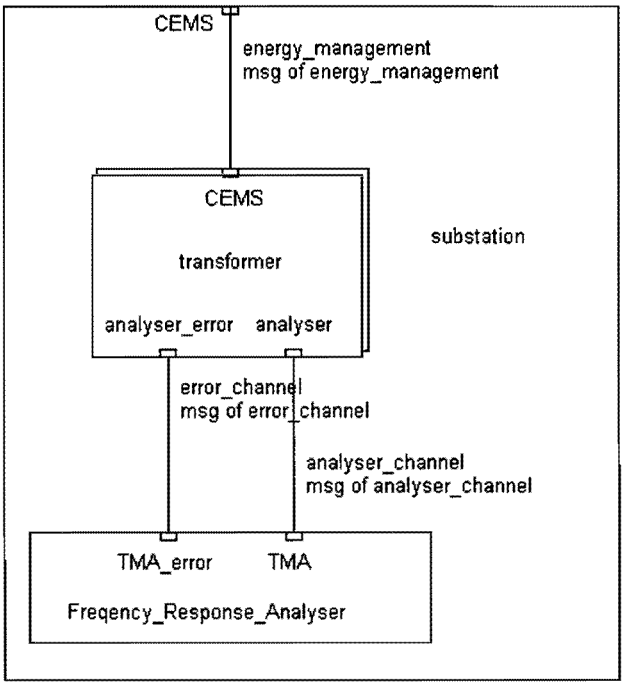


Figure 6.5 ISD of a substation

Figure 6.6 shows the ISD of the cluster class 'transformer'.

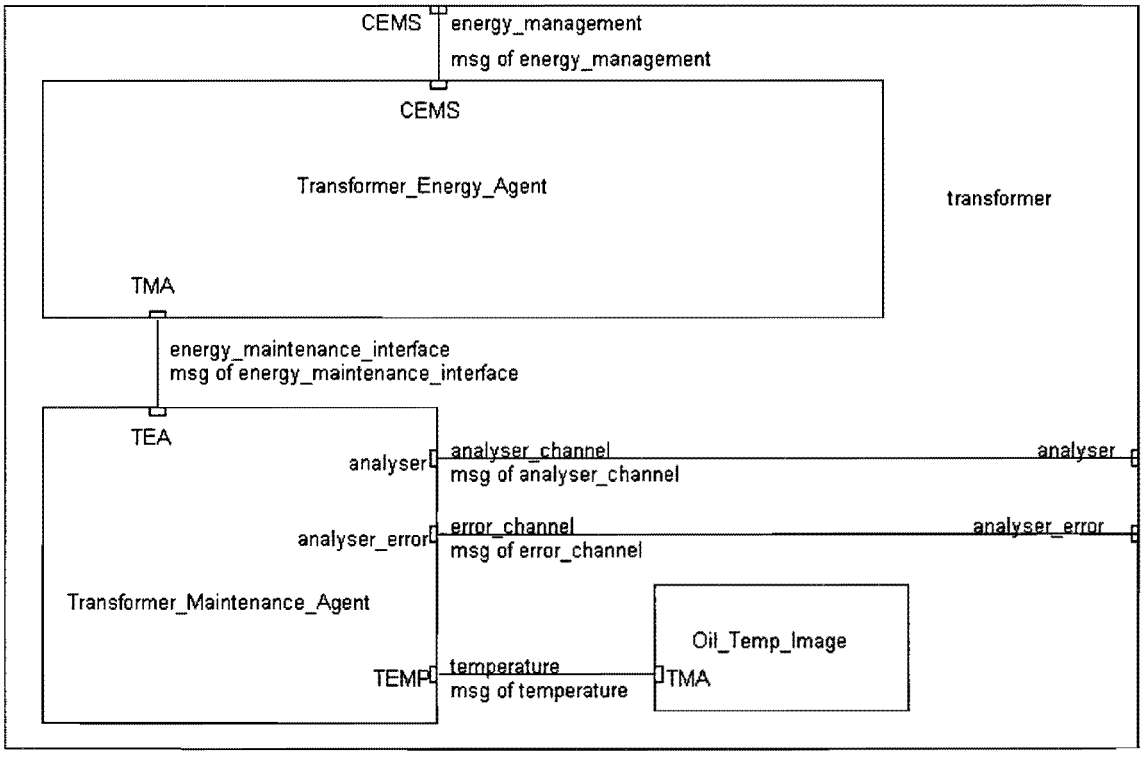


Figure 6.6 ISD of a transformer

6.4 Collaboration Model

This view on the system describes the communication between object instances. They communicate by sending messages. In the essential model, the messages from one object instance to another are more important than to find the structure of the communication channels that are used. Therefore an essential channel structure is used in the model. This gives us the liberty of defining a channel wherever needed for the purpose of communication. During development of the extended model, implementation details on the communication topology can be introduced and modeled.

Analysis of the system's application domain leads to different use cases, on the basis of which different scenarios of behavior can be defined. The scenarios are elaborated, after which for each process class all messages are identified. The behavior of the various process classes is being caught in the POOSL specification. The result of combining the scenarios' Message Flow Diagrams (see section 6.4.1) is the basis for the essential channels structure.

Sections 6.4.2 through 6.4.4 describe the chosen example scenarios, using a Message Sequence Chart (MSC), a narrative (which describes the scenario and explains the MSC), and a Message Flow Diagram (MFD), successively.
In the first place, a scenario 'normal' will be explained, explaining the usual message flow between objects, in case an analysis should be carried out in order to assess the condition of a transformer. In the ideal situation, everything goes by plan. This is the scenario that has been implemented in the simulation model.
In a realistic situation, however, a lot of thing may go wrong. A few failures have been integrated as behavior into the process classes of the model. Some of those failures are generated in the current model; others are not. In the latter case, the behavior that generates the occurrence of failures can be added later. The behavior that deals with the failures, however, is included and can be verified in a later phase.

6.4.1 Message Flow Diagram

Figure 6.7 gives the Message Flow Diagram of the modeled system. This diagram contains all messages that are currently used in the behavior description. The amount of messages hinders validation of the analysis. Not all messages are relevant to each scenario; the use of different scenarios limits the amount of information.
In the POOSL behavior specification (see appendix C), the 'normal' scenario includes all described failures. The POOSL behavior specification is programmed such that it implements a combination of the scenarios, mentioned below.

Usually, the values of some variables are transported via the parameters of messages. For the clarity's sake, parameters of the messages are not included into this diagram. Instead, they can be found in the interaction diagrams of section 6.5.2.

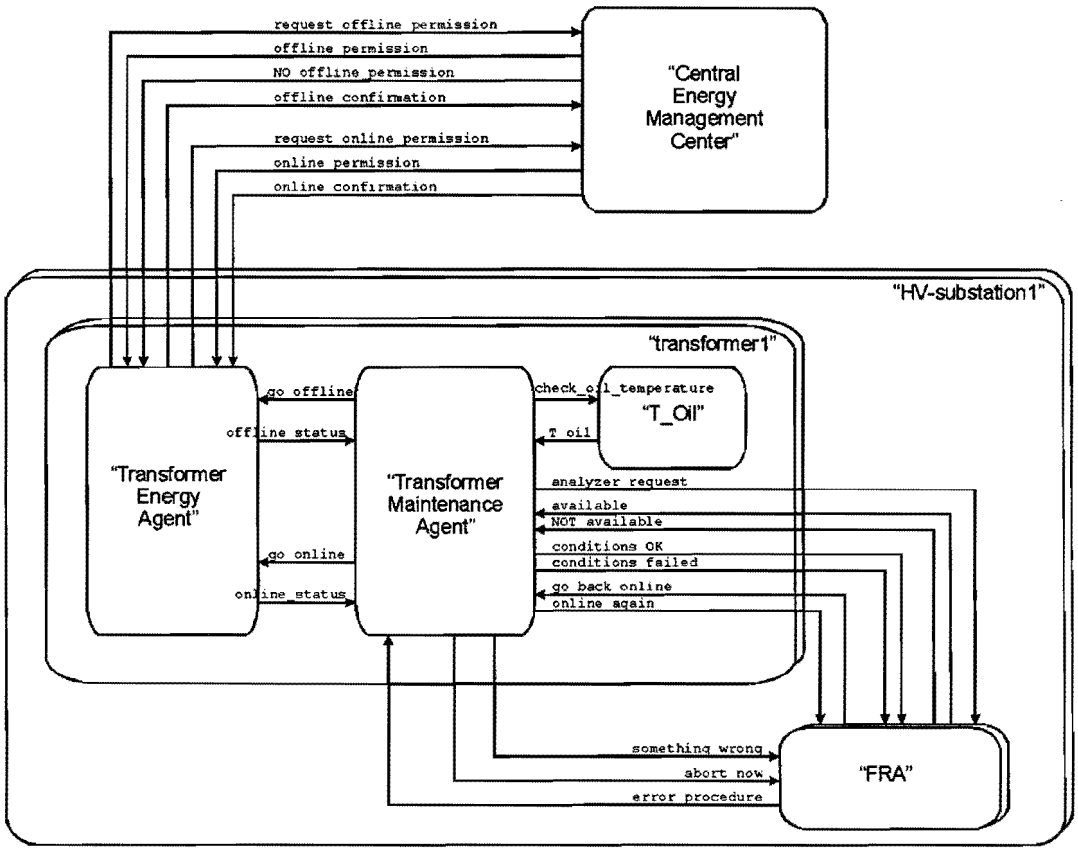


Figure 6.7 Message Flow Diagram of the modeled system

The sections below will describe a few of the scenarios. First, the *'normal'* scenario; secondly the scenario *'NO_offline_permission'*, and on the third place the *'conditions_failed'* scenario.
Finally, other scenarios are merely mentioned, but not worked pout in detail.

6.4.2 Scenario 'normal'

Various analysis tools are available in a substation on behalf of condition assessment. Depending on results of earlier measurements, the moment for performing a new analysis will be scheduled.
This scenario describes the ideal situation, when an analysis should be performed: permission is always granted, timeouts never occur and condition requirements are always met.

Message Sequence Chart

For easy reference (long messages), the messages in the Message Sequence Chart below have been numbered. The corresponding reference numbers will also be used in the narrative and MFD.

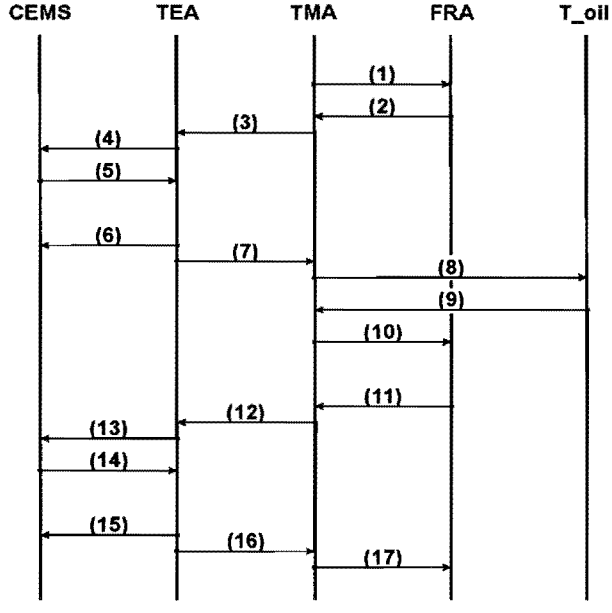


Figure 6.8 Message Sequence Chart of scenario 'normal'

Legend:
(1) analyser_request(component_ID)
(2) available(target_in_use, analyzer_ID, "offline, T_oil < 308K")
(3) go_offline(analyzer_ID, estimated_FRA_duration)
(4) request_offline_permission(component_ID, substation_ID, approx_duration, urgency)
(5) offline_permission(component_ID, substation_ID, "proposed_delay", proposed_delay, "maximum_duration", maximum_duration)
(6) offline_confirmation(component_ID, substation_ID)
(7) offline_status(offline_status, offline_problems)
(8) check_oil_temperature("Max temp:", max_T_oil)
(9) T_oil(temperature_status, temperature_problems)
(10) conditions_OK(component_ID)
(11) go_back_online(target_in_use, FRA_timer)
(12) go_online()
(13) request_online_permission(component_ID, substation_ID)
(14) online_permission(component_ID, substation_ID, "proposed_delay", proposed_delay)
(15) online_confirmation(component_ID, substation_ID)
(16) online_status(online_status, online_problems)
(17) online_again(component_ID)

Narrative

The TMA (Transformer Maintenance Agent) initiates the analysis by requesting (1) the analyzer (Frequency Response Analysis, FRA, in this case). FRA is available, will be allocated to the requesting component, and prescribes a few conditions that should be met, before the actual analysis can be started (2). In this case the transformer concerned should be offline, with an oil temperature below 308K.
The permission (5) to go offline is obtained from the Central Energy Management Center (CEMS), via TEA, the Transformer Energy Agent (3) (4). A confirmation (6) will be sent to CEMS, as soon as the transformer has been switched off, while the offline status is reported back to TMA (7).
The cooling down process starts. The maximum oil temperature is communicated (8). When the temperature condition is met, the object T_oil (oil_temperature) returns a message (9) (within a predetermined period of time).
The FRA starts measuring and analyses the measurement results, after the message (10) that all conditions have been met. When the analysis process finishes, the transformer will have to be switched on again (11). In advance, online permission should be obtained via TEA (12) from CEMS (13) (14), analogously to the offline permission. A confirmation to the CEMS (15) and the TMA (16) follows when the component is actually online again.
Finally, an online confirmation (17) makes the FRA directly available to other components again.

Message Flow Diagram

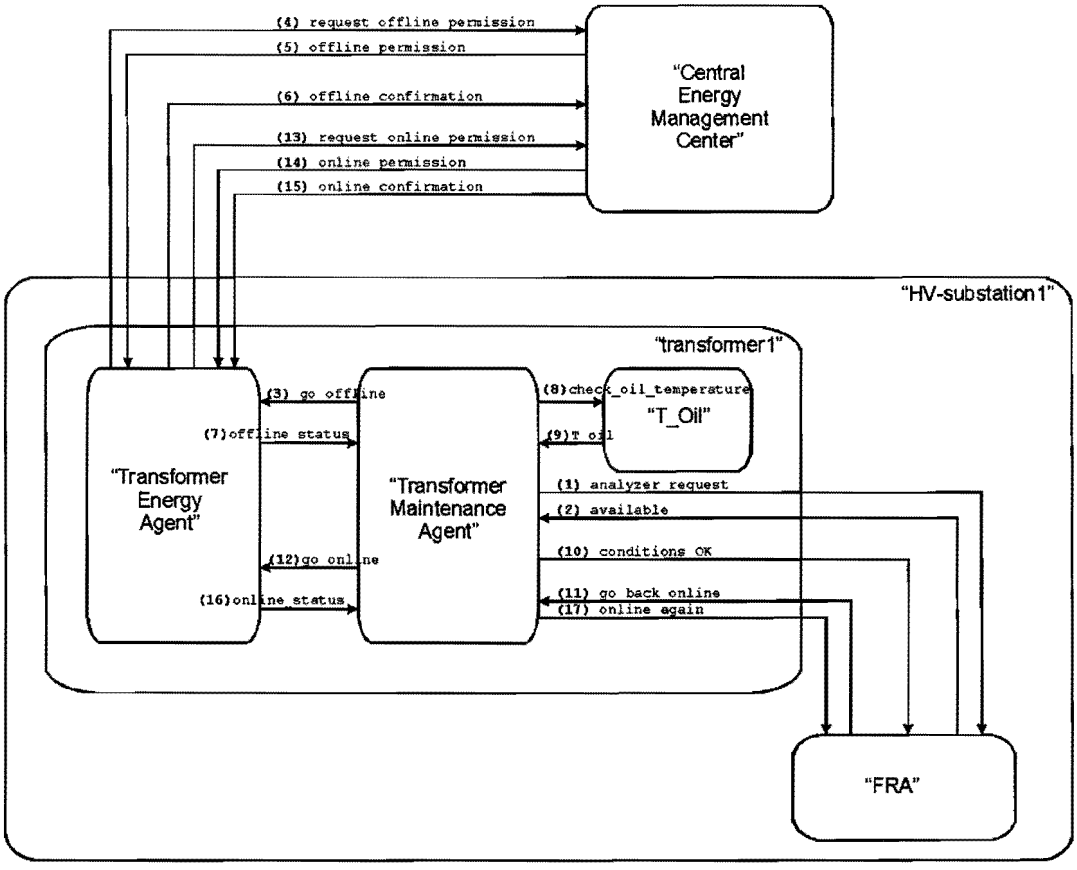


Figure 6.9 Message Flow Diagram of scenario 'normal'

6.4.3 Scenario 'NO_offline_permission'

This scenario shows the case that no offline permission can be obtained via the CEMS. This may be the case when other equipment is offline already, when the request is done.

Message Sequence Chart

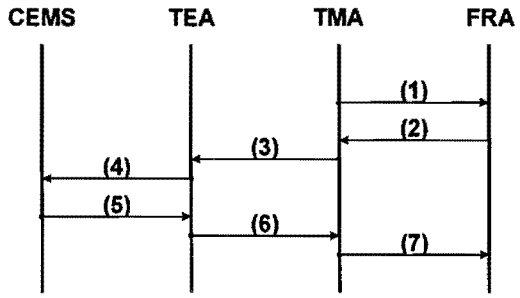


Figure 6.10 Message Sequence Chart of scenario 'NO_offline_permission'

Legend:
(1) analyser_request(component_ID)
(2) available(target_in_use, analyzer_ID, "offline, T_oil < 308K")
(3) go_offline(analyzer_ID, estimated_FRA_duration)
(4) request_offline_permission(component_ID, substation_ID, approx_duration, urgency)
(5) NO_offline_permission(component_ID, substation_ID)
(6) offline_status(offline_status, offline_problems)
(7) conditions_failed(component_ID, problems)

The variable 'offline_problems' of message (6) contains the value "no_offline_permission_by_CEMS", after which the TMA has to do a new request after a while.

Narrative

This scenario shows the case that the component does not get the offline permission by CEMS. Steps (1) through (5) are similar to corresponding steps in the previous scenario. The CEMS does not allow the component to go offline (6). As soon as FRA notices (7) that the conditions failed, the allocation to the component will be disappears and, consequently, the FRA's internal state is reset.

Message Flow Diagram

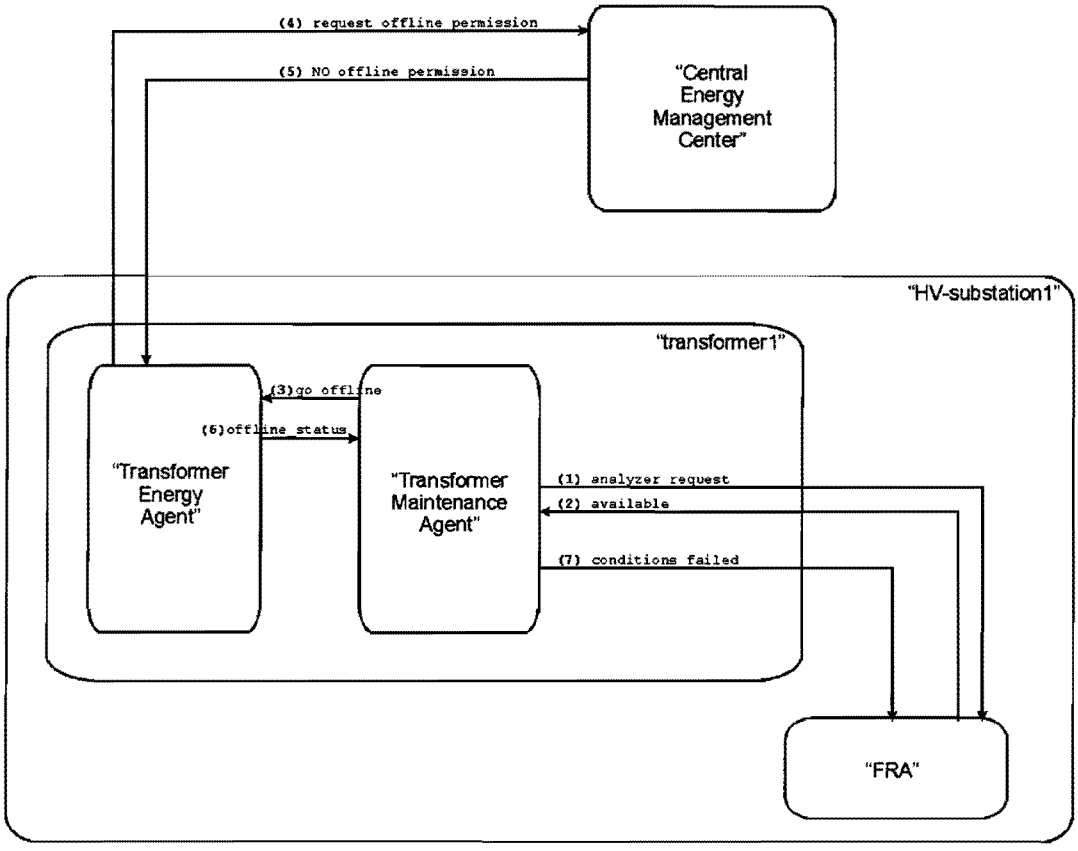


Figure 6.11 Message Flow Diagram of scenario 'NO_offline_permission'

6.4.4 Scenario 'analyzer in use'

This section shows the competition for the FRA between transformers, as only one FRA can be performed at the same time.

Message Sequence Chart

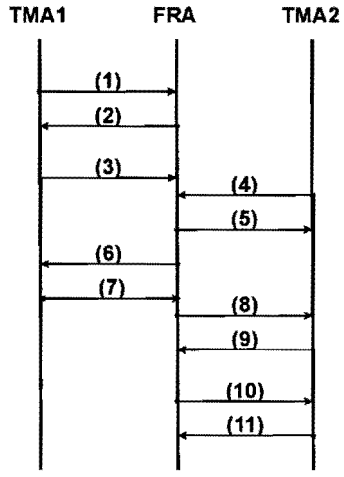


Figure 6.12 Message Sequence Chart of scenario 'analyzer_in_use'

Legend:
(1) analyser_request("transformer1")
(2) available("transformer1", analyzer_ID, "offline, T_oil < 308K")
(3) conditions_OK("transformer1")
(4) analyser_request("transformer2")
(5) available("transformer2", analyzer_ID, "wait")
(6) go_back_online("transformer1", FRA_timer)
(7) online_again("transformer1")
(8) available("transformer2", analyzer_ID, "offline, T_oil < 308K")
(9) conditions_OK("transformer2")
(10) go_back_online("transformer2", FRA_timer)
(11) online_again("transformer2")

Narrative

Because there is only one analyzer available in each substation, and many transformers, the analyzer gets allocated to one transformer (with message (3)). However, the analyzer is able to remember a second request. If the FRA is in use while a second request (4) is done, the identity of the second transformer will be saved in a buffer.
When transformer 2 receives an 'available' message (5) with "wait" as condition, it waits for another 'available' message.
After the analysis on the first transformer has finished, after (6) and (7), the FRA sends an 'available' message (8) to the second one, with the usual conditions for that analysis tool (offline, T_oil < 308K). The second transformer is being analyzed and eventually goes back online again (9) and releases FRA (10).
The interaction with TEA, CEMS and T_oil is not included in this scenario, but goes like in the normal scenario. We abstract from that in this section.

Message Flow Diagram

In this image, all irrelevant parts are omitted. Only the FRA and the TMAs of the transformers involved are important for this scenario.

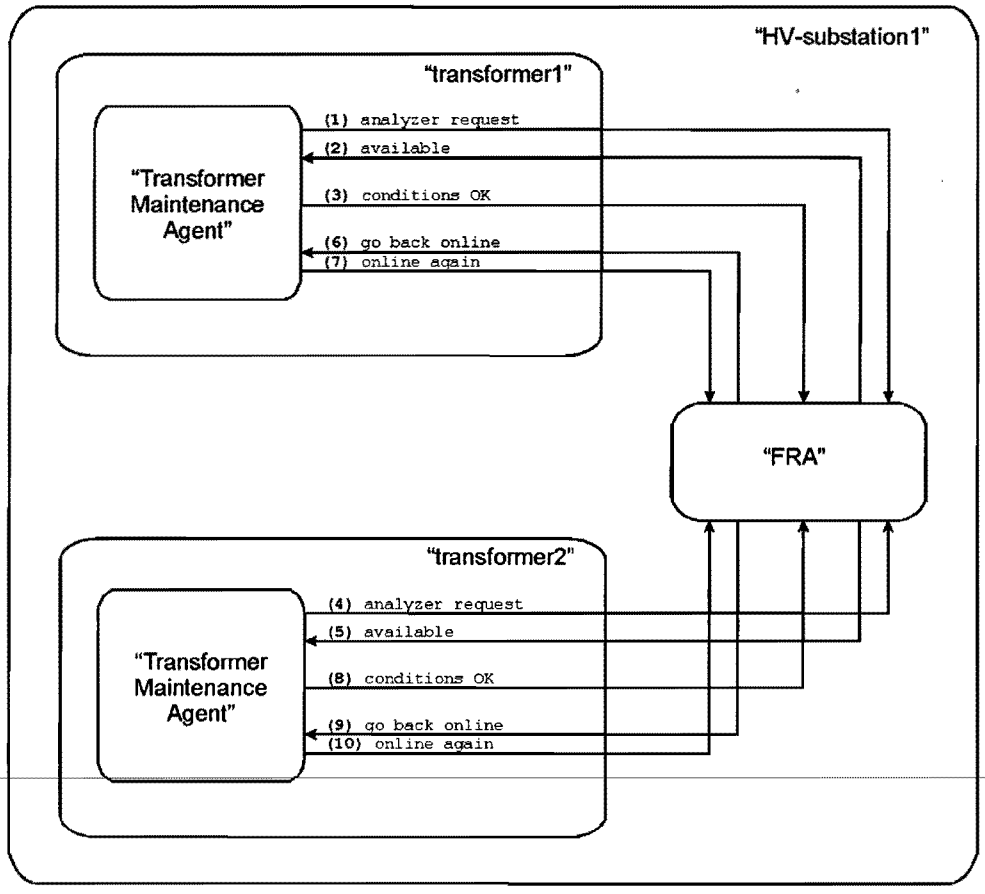


Figure 6.13 Message Flow Diagram of scenario 'analyzer_in_use'

6.4.5 POOSL Description

This section gives a short description of behavior specification in POOSL.
The environment shown in Figure 6.14 shows the environment in which the behavior of a process class is being specified.

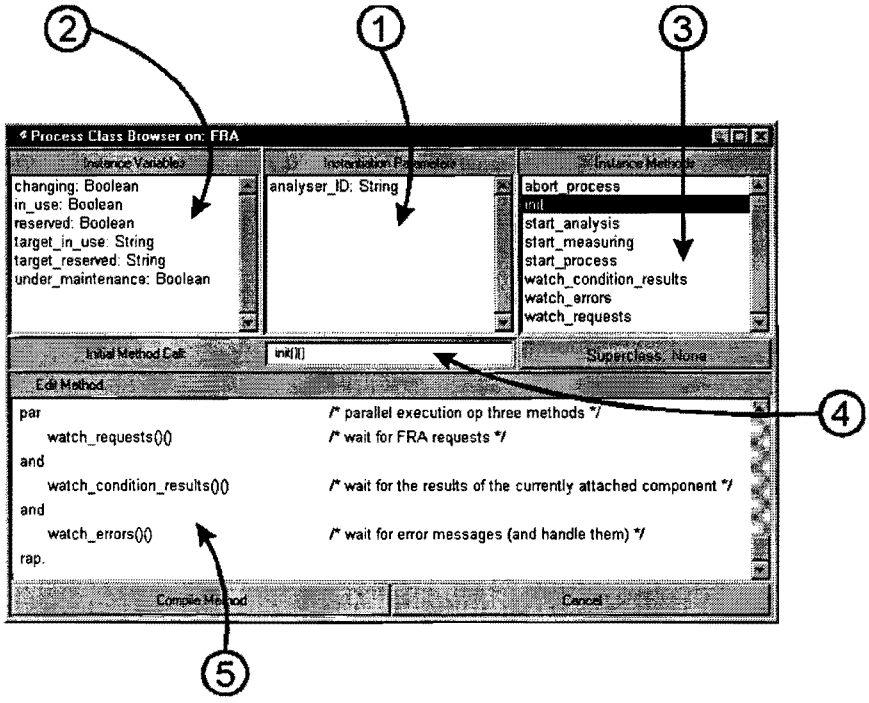


Figure 6.14 Process Class Editor

Global variables (either variables or parameters) can be added in (1) or (2), respectively. They are characterized by a name and a type. The ones that are shown in the picture have been chosen in an earlier stage.
The instantiation parameters (1) allow all instances of this class to use distinct parameter values. In this example, the identity of the analyzer has been caught into a variable 'analyser_ID' of the type 'string'.
Instance variables (2) are global variables within the process class, and can be altered by any of the methods, even (quasi-)simultaneously. For example, the Boolean variable 'in_use' indicates if the analyzer is directly available for analysis, and is an *attribute* of the class 'FRA'.

The behavior specification (5) can be created or edited in the lower part of the screen. If new behavior is added to a class, the corresponding method name will be show up (after compiling) at the top right corner of the editor, which lists the methods (3) of the process class.
The method that should be called on startup can be chosen in the initial method call field (4). When the simulator starts executing the behavior, in this example, the first method to be called is *init ()()*[1].

[1] The input and output parameters of a method can be entered between the two pairs of parentheses, respectively

The behavior that is captured in each method can be edited later on by selecting the method concerned in (3) and changing the POOSL code.

The parallel-statement `par...and...rap` is one of the most important statements in POOSL, which makes parallel behavior possible.
As can be seen, *init ()()* contains parallel operations. The three methods *watch_requests()()*, *watch_condition_results()()* and *watch_errors()()* are called (and executed) simultaneously. None of the mentioned methods uses input or output parameters and are given empty pairs of parentheses.
For further details on the language POOSL, we refer to 0 or [17].

Another statement that could use some explanation is the select-statement `par...or...rap`. This function will be illustrated, using the example below.

```
...
sel
   TMA!available(target, analyser_ID, "offline, T_oil < 308K");
   ...
   ...
or
   TMA!NOT_available(target);
   ...
   ...
or
   delay 30
   ...
   ...
les;
...
```

At the moment the 'select' statement is executed, the simulator selects the first branch that can do a 'transition'. Of each branch, only the first statement will be taken into account.
In the example, either the **transmission of 'available(...)'**, or the **transmission of 'NOT_available(...)'**, or a **delay** is the first one that can make a transition.
If no message could be sent (i.e. no receiver is prepared to accept the message - rendezvous communication), the delay is the first statement that can and will be executed. The process continues with the statements within 'delay'-branch.
After the last statement of the 'delay'-branch has finished, the next statement that will be executed is the one directly below 'les;'.
If a rendezvous communication can be achieved before the delay time has elapsed, SHESim will select and execute the branch that starts with the mentioned rendezvous communication step.
This structure has been used several times as a watchdog in the example model. Timeouts can be handled in this way.

6.5 Simulation and Verification

Before continuing this section, a remark should be placed on the cluster class name 'EHV_substation'. Although the term 'HV-substation' is used throughout the entire report, the simulation model still uses the 'old' name for this cluster. However, this should be changed into 'HV_substation', as the EHV-substations are not part of this example.

6.5.1 Simulation

Next to the *behavior* of the distinct process classes, which can be specified in the process class editor, its *structure* can be created in the 'System Level Editor' of SHESim.

Figure 6.15 shows the simulator window in SHE.

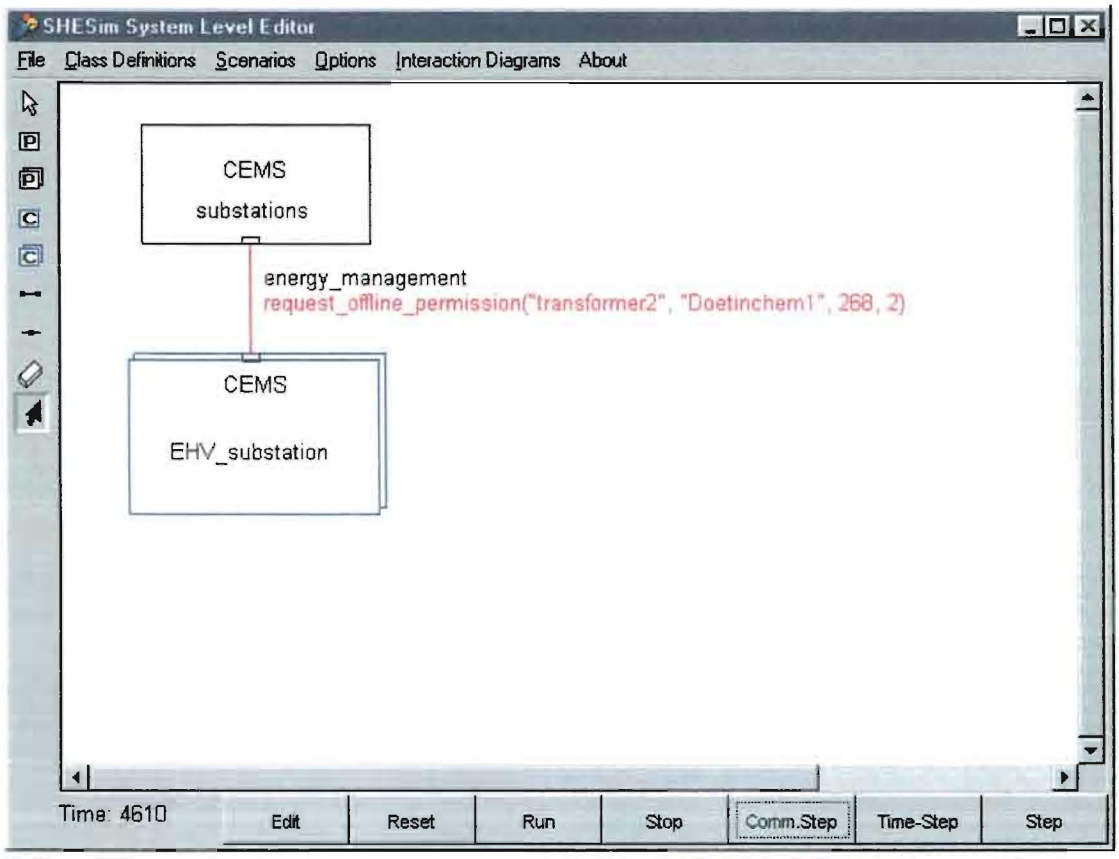


Figure 6.15 SHESim simulator window

The buttons on the left side can be selected for creating and editing the structure of the system, as well as the structures of cluster classes, as graphics of processes, clusters and channels.
From top to bottom, the buttons activate the **selection** tool, the **process object** tool, the **multiple process** tool, the **cluster** tool, the **multiple cluster** tool, the **channel** tool, the **node** tool, the **erase** tool and a second **selection** tool[1], respectively.
The example in the picture above contains one instance of the process class 'CEMS' and multiple instances of cluster[2] class 'EHV_substation'.

Note that the cluster class editor has the same look as the system level editor, with the exception that the cluster class editor has an additional **port** tool button, which allows the creation of a port to the boundaries of the cluster.

The buttons below are part of the simulator.

[1] The first selection tool allows editing of the selected, the second one does not (e.g. during simulation)
[2] The difference between process and cluster classes instances is shown by the color of its boundaries: black for process objects, blue for clusters

Their functions are to: change back to edit mode (**Edit**), reset to time 0 (**Reset**), execute behavior continuously (**Run**), pause simulation (**Stop**), run until a message is communicated visibly (**Comm.Step**), run until the next time change (**Time-Step**) and finally step by step through the simulation (**Step**).
The currently sent message is displayed in red.

During simulations, the internal structures and messages of cluster classes can be inspected in more detail (by right-clicking the target object and selecting the 'inspect' option).
Two examples of cluster class inspections are shown in Figure 6.16 and Figure 6.17. These inspectors also display the active messages in red.

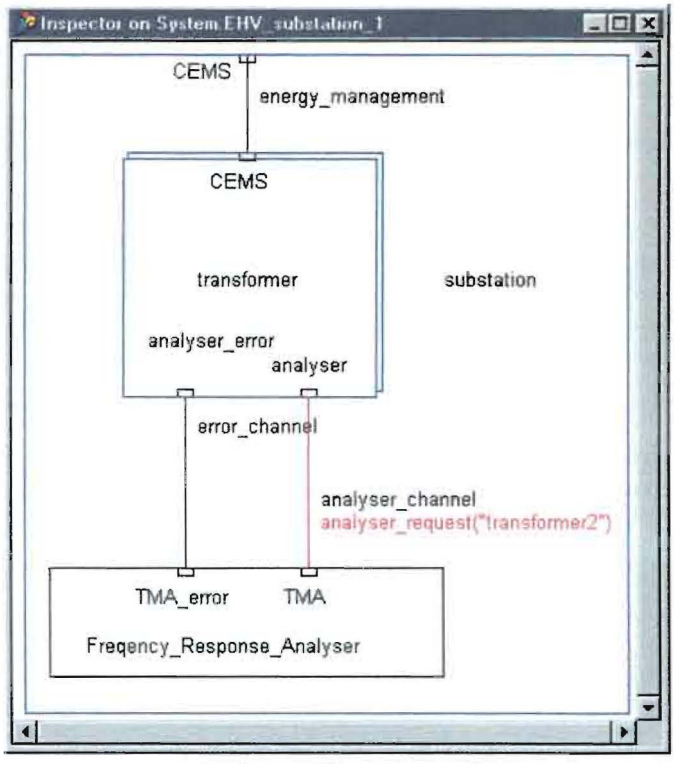


Figure 6.16 Inspector window on cluster class instance 'substation'

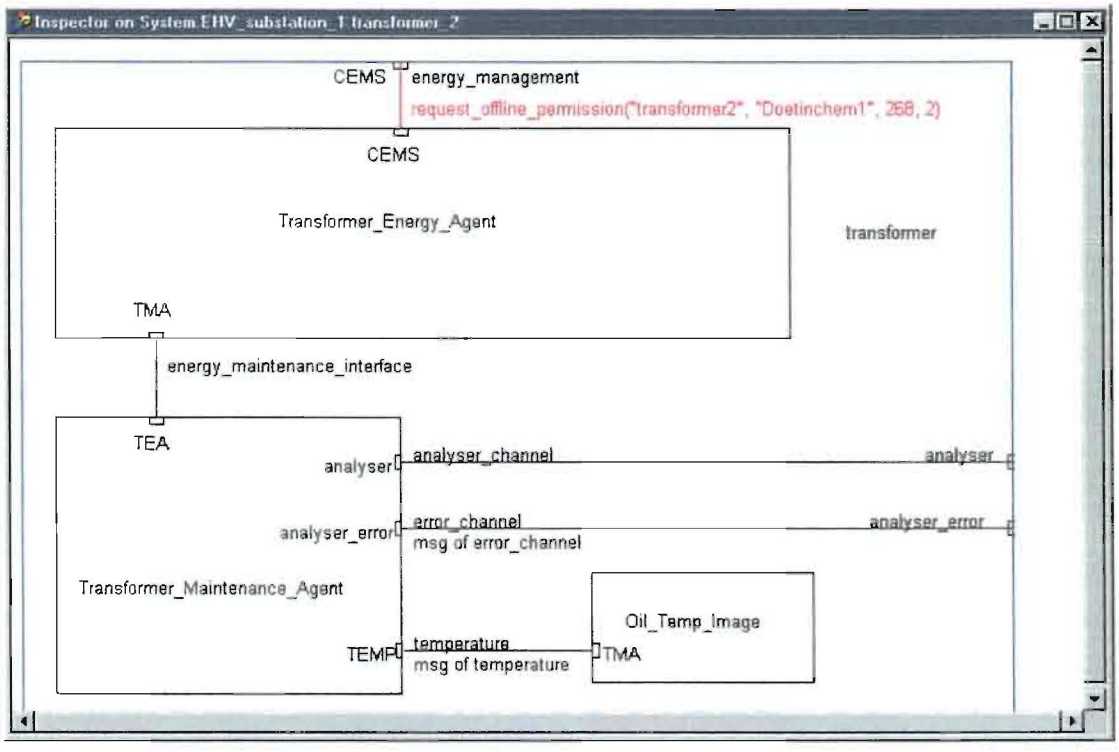


Figure 6.17 Inspector window on cluster class instance 'transformer'

It is also possible to inspect a process class, during simulations. The inspector window of 'TEA' is like in Figure 6.18 below. Various variables can be monitored, along with the active statements and the selected action.

Additional information on the simulation environment can be found in [17].

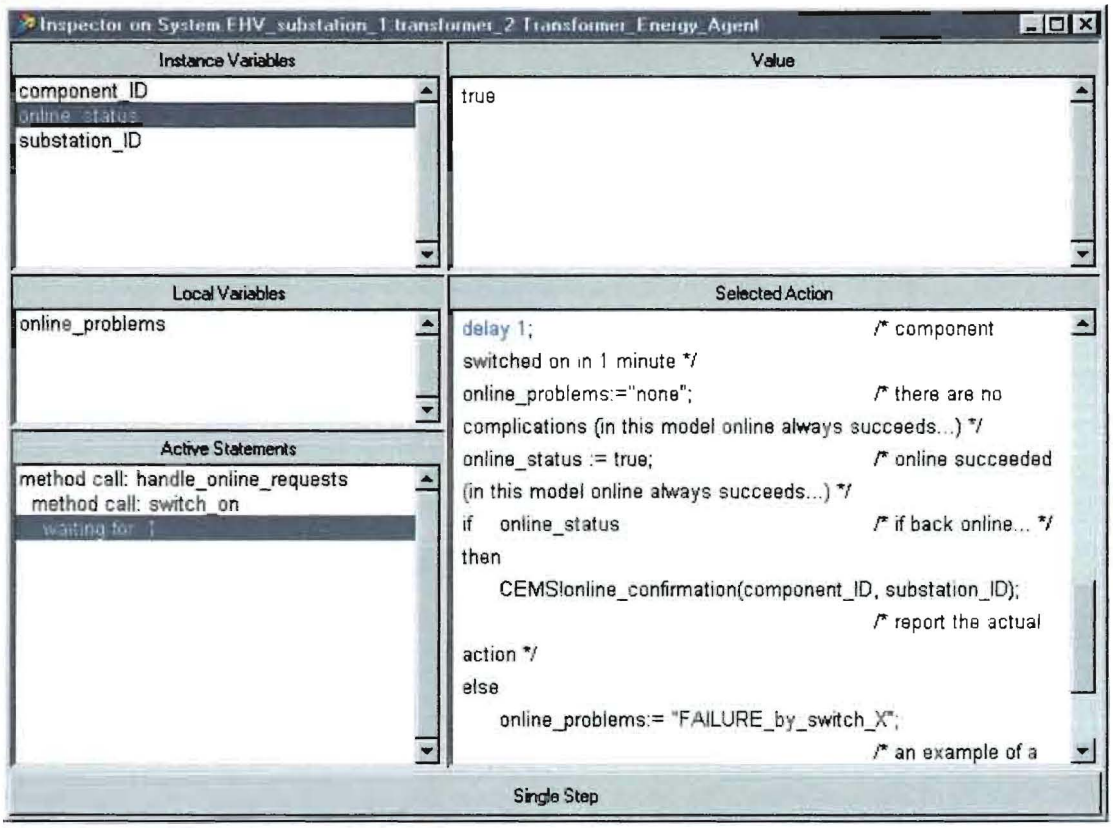


Figure 6.18 Inspector window on process class instance 'TEA'

These features are of great help in the verification process of the model.

6.5.2 Verification

The behavior of the various object classes has been verified during the process of modeling and simulating. The resulting POOSL file that contains the behavior and parameters of all classes is included in appendix C. A few examples of Message Sequence Charts, generated by the simulator, can be found below (Figure 6.19 through Figure 6.22 on several levels of detail).

The first interaction diagram (Figure 6.19) shows the communication on system level, between 'CEMS' and the substations. The communication is confined to obtaining offline and online permission. The two substations randomly send offline requests. In some cases, as a result of another transformer being offline already, the CEMS rejects a request by responding: 'NO_offline_permission(...)'.
For example, the two first rejections regarding EHV_substation1 are based on the fact that EHV_substation2 is offline at that moment. Little time later, all equipment goes online again. When EHV_substation2 sends a new request (at time stamp 8790), permission is granted.

The second screenshot (Figure 6.20) shows the competition of two transformers for the FRA within a substation. The message 'NOT_available(...)' will never be generated in this model, because there are only two transformers in each substation

of the model. Every second request can be scheduled (reserved). If the number of instances increases, the possibility of the analyzer not being available increases.

At time stamp 21973, a second request is done, while the analyzer is in use. This means the request is to be scheduled. The corresponding 'available(...)' message contains the condition 'wait'. After the first analysis has finished (time stamp: 22089), the second transformer receives a starting signal via the message 'available' with the usual conditions, instead of "wait".

The message flow and sequence within a transformer is shown in Figure 6.21, focusing on the interaction between TEA and TMA.

The final screenshot (Figure 6.22) shows the interaction at several levels at once.

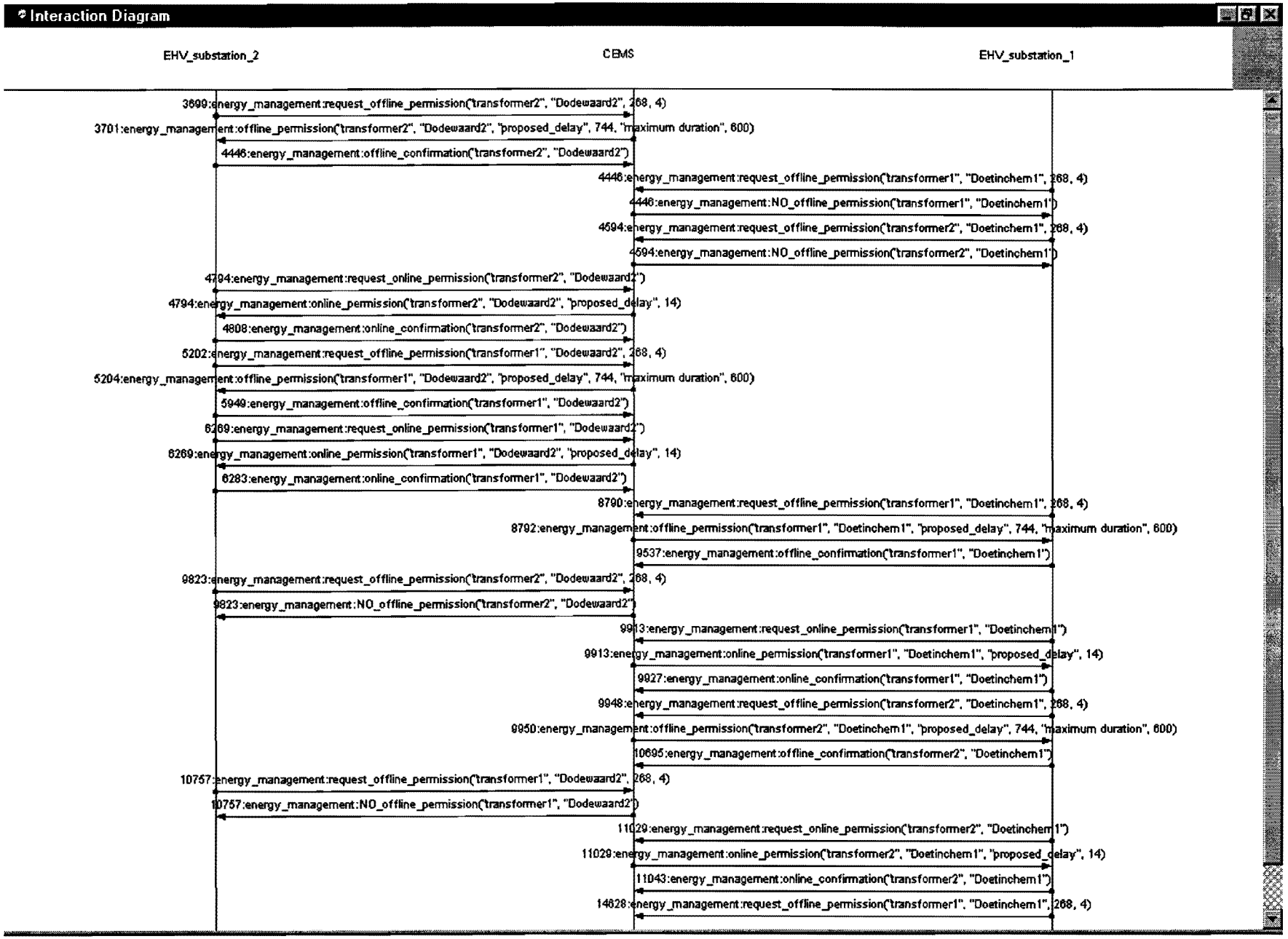


Figure 6.19 Interaction Diagram on system level

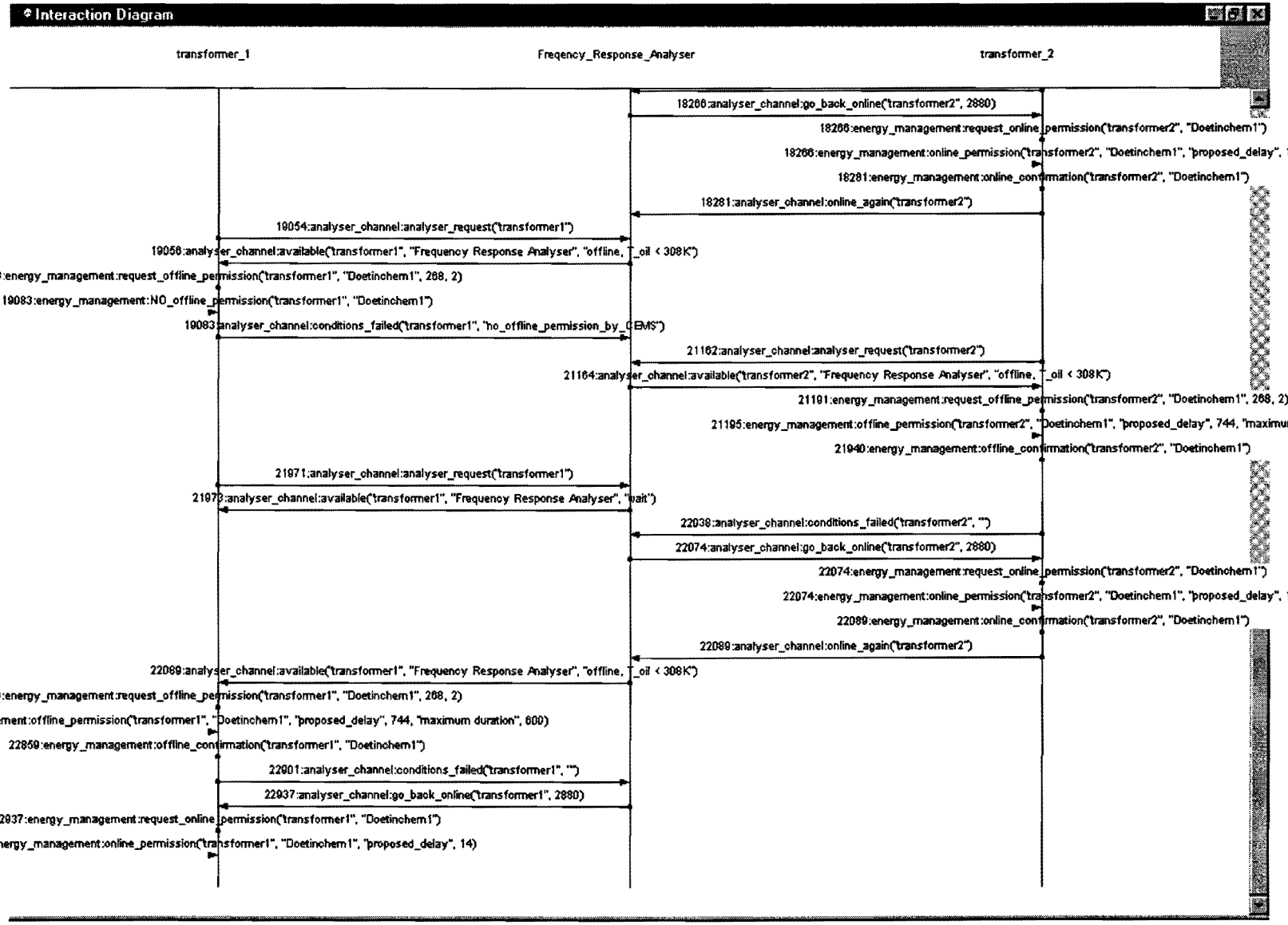


Figure 6.20 Interaction Diagram within substation

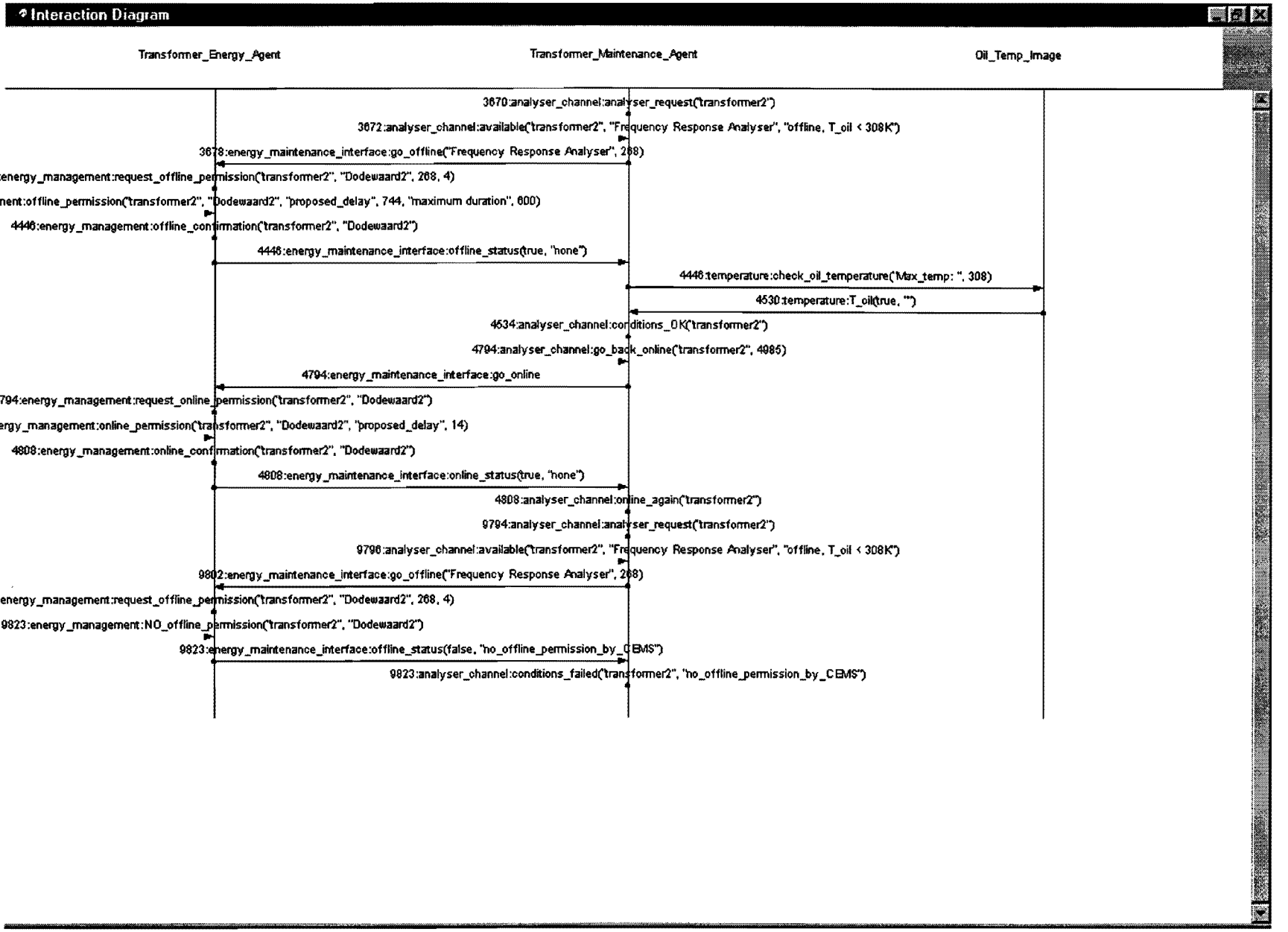


Figure 6.21 Interaction Diagram within a transformer

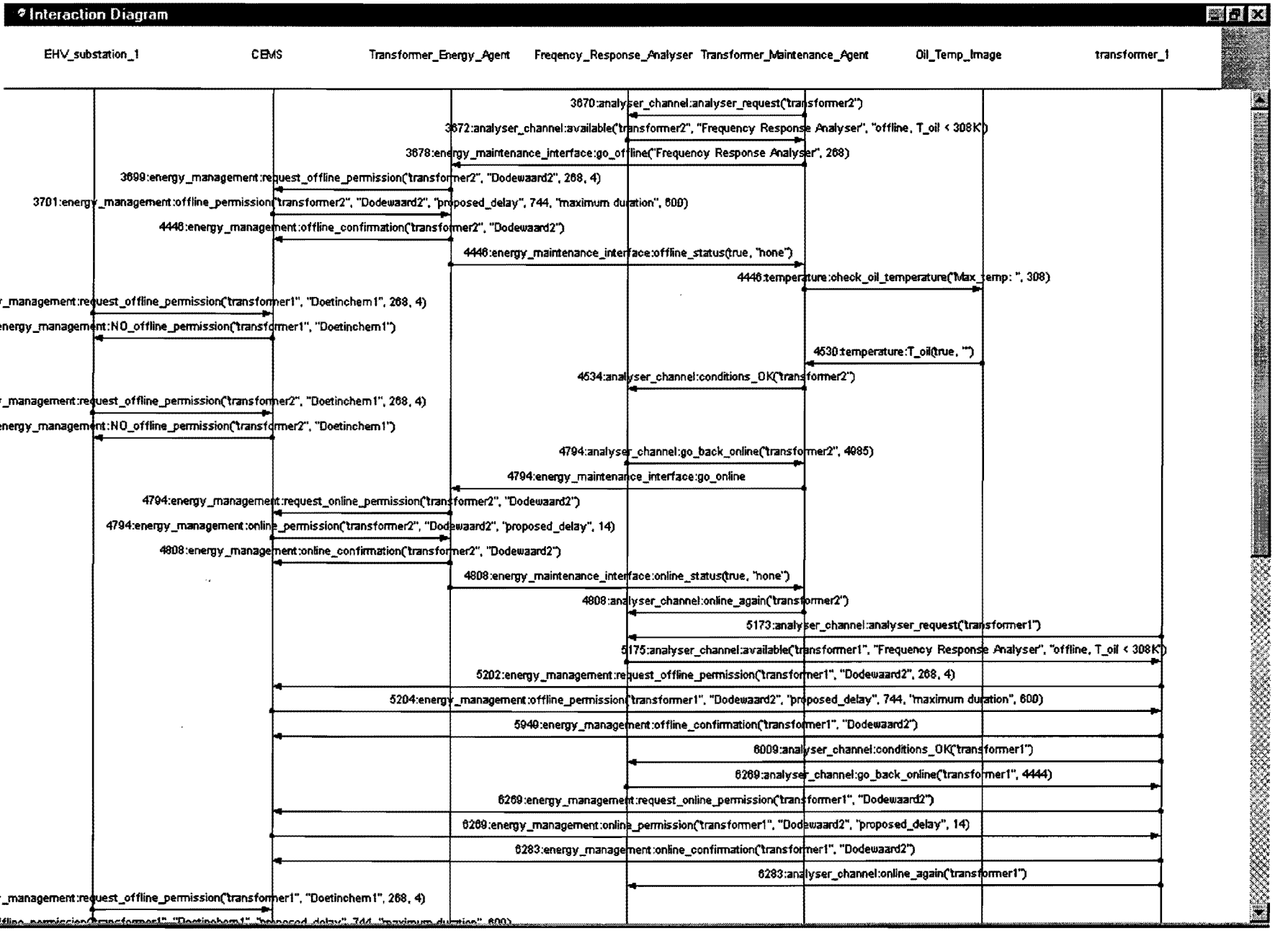


Figure 6.22 Interaction Diagram of the entire system

6.6 Experiences with SHE

6.6.1 Process

To come to this example model, starting from a number of scenarios defined, the first step has been to model the necessary message exchange between the distinct object instances. (Note: no 'multiple instances' of substation or transformer yet.)
After most of the bare communication worked properly, parallel operation has been introduced. Tasks that have to function at the very same time within a process class are executed via the '`par ... and ... rap`' statement. Time and delays have also been introduced in this stage.
At this moment the single substation with one transformer is being replaced by multiples. This also required some adaptations to the message reception (see also next section).
After a number of iterations, during which the simulator aided in finding errors and inconsistencies, error-handling procedures have been added. Starting from the ideal situation in scenario 'normal' of section 6.4.2, provisions for handling possible failures have been made, like handling timeouts, simultaneous requests and reacting on denial of offline permission request (from the CEMS).
'Sabotaging' behavior has been included into the behavior of the process classes, in order to generate some of the expected failures. The mechanisms that handle these failures can be verified in this way.
Random generators are helpful to simulate 'real' behavior and to avoid regularities.

6.6.2 Encountered Problems and Solutions

During the process of modeling, a lot of problems have been taken care of. This section mentions some of those and briefly explains the chosen solution.

Initiative questions have risen several times. Some processes can be initiated by multiple instances, which means a sensible choice has to be made on that.
An example is the question who takes the initiative of carrying out a new FRA. The answer (in this example) is that the TMA of a transformer takes the initiative to do an FRA-request. The reason is that the transformer should know its current condition and should know when the next measurements are scheduled. The FRA is a shared device that 'just' carries out an investigation for any of the transformers within the substation.

Another example is found in the process class 'T_oil'. At a certain moment, a message is received from the FRA. One of these parameters is the maximum oil temperature. T_oil starts measuring and as soon as the actual oil temperature meets the requirements (below maximum), it sends a message to TMA. Meanwhile the TMA may run other tasks rather than waiting for this response.

A third case covers the buffer-function in FRA as shown in section 6.4.4. After the first analysis has finished, the FRA initiates the sending a message to the device, whose request has been scheduled. An alternative decision could be to let the 'second' device do a new request, laying the initiative at the side of the device, instead of the FRA.

The benefit of my choice is that the FRA will be carried out **directly** after FRA becomes available again, which will not be likely in the alternative case (as the TMA does not know the FRA timing).

Another problem I addressed during modeling was *simultaneously changing global* variables in a process class. A number of global variables can be used simultaneously in two parallel methods. Each method may need to change a number of variables 'at once' on the basis of their value. 'At once' means that all variables need to be changed at the same time, due to a certain event (with no delay!). However, the simulator may change global variables within zero time, but actually, the transitions of the two 'parallel'[1] methods are made sequentially. Which statement will be carried out after another one, will then be chosen non-discriminatory, by the simulator.
A similar example is shown below:

```
if  x<4
then
  y:=2;
  z:=x;
  x:=3
else
  x:=3;
  y:=4;
  z:=9;
fi;
```

If two of these methods simultaneously try to change the variables, problems may arise. The changing of a variable in one method may influence the process of the other one.

For example: if x = 0, initially, then Method1 evaluates the data expression x<4, which turn out to be true, and changes x, y and z to 3, 2 and 0, respectively. However, if the second method changes x meanwhile, the resulting values of method1 will not be correct.

To avoid these problems, the variables concerned should only be changed simultaneously. No interference of another method is allowed until all variables have been changed. The problem was solved by the creation of an additional global variable 'changing', which indicates that the set of variables is being changed at the current moment.
Both methods included the same 'guard' command:

```
[changing not] changing:=true;
if  x<4
  .
  .
  .

fi;
changing:=false;
```

The first method (that wants to change the variables) checks 'changing' (which is still false). The variable 'changing' is set to true at exactly that moment, blocking the access for the other method.

[1] Actually, 'quasi-parallel'

If all the changes have been made, 'changing' is reset to false again. Now, the second method has access again...

Later, after finishing the example simulation model, the 'atomic' command { ... } appeared to guarantee the execution of all statements inside the pointed brackets at once, with no interference of other parallel methods. However, this option was not mentioned in my POOSL manual, and therefore not applied.

6.6.3 Future development

Taking the model in this chapter as a starting point, further developments are to be made. This section gives some possible directions.
First of all, the choices of scenarios and object classes should be verified, in the application domain. A few adjustments may be required, as a result of feedback.

This small model could be extended, in the first place by adding databases. This may be on several levels, like NMC level and substation level. Alternatives on the location of certain information should be mentioned and evaluated.
The Central Maintenance Management Center (CMMS) has not been used in this model yet. However, the 'real' maintenance functionality on system level, is important for modeling the total MICS.

What could be useful is to model the current Energy Management System (EMS) in POOSL, together with the communication protocols of the utilities industry (IEC870-5).

Chapter 7. Conclusions and Recommendations

7. Conclusions and Recommendations

7.1 Conclusions

Due to liberalization of the energy distribution sector, electricity companies need to reduce costs. New technologies will have to be explored, in order to achieve the cost reduction and possibly extending the networks' capabilities.
Remote Monitoring (RM) is a technique, which can aid in the realization of Condition Based Maintenance (CBM). Beyond this, RM may also increase the functionality of the network by the adding new features.

The application of RM extends the communication facilities of electrical equipment in the field, and makes the step towards autonomously functioning networks easier. Next to the potential benefits, there are several points of attention. Recent studies may be used keep oneself informed about identified problems and their possible solutions.

The use of the IEEE methodology for the collection and organization of requirements has been experienced. The process will even be done more justice during a follow-up project, in which a multidisciplinary team dives into specification of the system. The resulting document (System Requirements Specification or 'SyRS') will prove to be a solid basis for future studies.

The modeling technique described in this report, proved to be a very powerful tool in the phase of specification. What makes SHE so valuable is the early stage in which problems, options and alternatives can be identified, by focusing on various aspects. Moreover, the methodology covers the entire design phase, up to implementation.

The Object Oriented (OO) philosophy may be an obstacle for those who are not familiar with these techniques. However, once the essence of Object Oriented Modeling has become clear, its benefits can be fully exploited. With its close connection to the application domain, OO-modeling provides better understanding of requirements.

A benefit of the high abstraction level is that the generic analysis approach can also be applied to for the development of other systems, not necessarily bounded to the domain of electrical utilities.

Considering the complexity of systems' design on the one hand and the scale of the target system on the other hand, an investigation like this has appeared to be very complex to be handled by a single person. The complexity and the high level of abstraction of design methodology should be taken into account when continuing the project.

Eventually, the real benefits must appear from a pilot project. Results can then be used to convince potential customers of the system's benefits. KEMA-TDP might consider offering consultation, assisting interested companies with development, integration, customizing and migration.

7.2 Recommendations

For continuation of the project, I propose the following work breakdown:

Potential customers can now be approached for their collaboration with the proposed pilot. KEMA should find an appropriate partner in industry, like e.g. one (or more) energy distribution companies. This partner is the key to important input from years of experience in practice.

Limitation to electrical energy distribution sector is not strictly necessary. Contribution of the gas distribution sector will be valuable as well, considering the many similarities between the two utilities. The resulting project will have a more generic and company-independent character, added with some specific functionality.

A multidisciplinary team is to be created, containing members of at least the following disciplines:

> Customer (e.g. an energy distribution company)
> As mentioned above, this includes companies that own and manage similar large-scale networks. Important is the added value of remote monitoring, compared to the existing situation.
>
> Information and communication technology (ICT)
> Regarding the fact that the system's design includes aspects of information and communications technology, the need for ICT-experts will be evident. Some experts will give input and feedback on communication techniques; others concentrate on information technical aspects, e.g. how to manage a historical database. This group of experts also includes modeling experts, who cannot be missed.
>
> Maintenance experts
> In the first place, the system will concentrate on maintenance. Both operational and strategic maintenance experts can provide the necessary (detailed) information on various maintenance issues. People 'on the working floor' can often add very important practical issues, while to maintenance managers the decisions on maintenance strategies may concern.
>
> Coordinating institute
> Finally, the last party the team should contain, is a coordinating one. Being an independent research institute, KEMA may be the most dedicated party to fulfill this task. The internal knowledge at KEMA's will probably be used during the development process.

Active participation and commitment of every team member are necessary conditions.

After the birth of a team, every team member should be familiarized with the methodology of requirements capturing. They also need to have background information on the essential aspects of the modeling technique. Beside that, the modeling expert should be fully familiar with the SHE methodology, its capabilities and its limitations.

Effort should be taken into the composition of a set of initial requirements. As section 4.1 shows, many techniques can be used for requirements capturing, like e.g.

several brainstorm sessions. Initial models (including simulations) should accompany these requirements.

Consecutive brainstorm sessions may be necessary to evaluate the results at that moment, after which the acquired feedback is used for adjustments of requirements and models.

A few adjustments may have to be made to the initial pilot project proposal, on the basis of a semi-permanent specification document. The next step is to execute and evaluate the pilot. The system will be tested and evaluated, in order to eliminate design flaws and optimize the system in its environment.

A successful prototype system should lead to the development of the 'complete' MICS, after which integration and application of the system follows.

References

[1] Aicklen, G.H. and P.M. Main
REMOTE CONTROL OF DIVERSE NETWORK ELEMENTS USING SNMP.
In: MILCOM 95. Universal Communications. San Diego, 5-8 Nov. 1995.
New York: IEEE, 1995. Vol. 2, p. 673-7.

[2] Andersson, E. and T. Sondell, P. Mäenpää, O. Vähämäki
INTEGRATED REMOTE CONTROL AND PROTECTION SYSTEM WITH DISTRIBUTED CONTROL AND PROTECTION UNITS.
In: CIRED . 10th International Conference on Electricity Distribution. Brighton, 8-12 May 1989.
London: IEE, 1989. No. 305, p. 331-40.

[3] Antal, M.
BEHEERSASPECTEN VAN DE ENERGIEVOORZIENING EN TELECOMMUNICATIENETTEN.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, August 1992.
EUT Report 5756.

[4] Apostolov, A. and P. Gregory
NETWORK INTERFACE MODULES FOR MICROPROCESSOR RELAYS INTEGRATION INTO A SUBSTATION AUTOMATION SYSTEM.
In: Sixth International Conference on Developments in Power System Protection. Nottingham, 25-27 March 1997.
London: IEE, 1997. No. 434, p. 309-12.

[5] (various authors)
In: symposium STATIONSAUTOMATISERING (de digitalisering van de secundaire installaties in elektrische netstations
Eindhoven, April 2, 1992. Ed. by J.F.L VanCasteren.
Eindhoven University of Technology, Draaistroomdispuut WALDUR 1992 (P.O. Box 513, NL-5600 MB, Eindhoven, Tel. +31 (0) 40 247 42 64, Fax. +31 (0) 40 243 43 64). Pages: ###.

[6] Ball, R. and K. Hill
A GENERIC COMMUNICATIONS LANGUAGE-FOUR YEARS OF EXPERIENCE.
In: Sixth International Conference on Developments in Power System Protection. Nottingham, 25-27 March 1997.
London: IEE, 1997. No. 434, p. 304-8.

[7] Boer, R.A.
IMPLEMENTATIE TOESTANDSAFHANKELIJK ONDERHOUD.
Den Haag: ENECO Den Haag/Voorburg, afdeling Infrastructuur, 1998.

[8] Bolam, D. and J. Robertson, R. Stirling, F. Baldinger
EXPERIENCE IN THE APPLICATION OF SUBSTATION CO-ORDINATED CONTROL AND PROTECTION WITH DEVELOPMENT TRENDS IN THE STANDARD CONTROL SYSTEM OPEN ARCHITECTURE.
In: Fourth International Conference on Power System Control and Management. London, 16-18 April 1996.
London: IEE, 1996. No. 421, p. 92-7.

[9] Bollström, T. and S. Pajukoski, S. Pettissalo, T. Hakola
SUBSTATION SECONDARY EQUIPMENT SYSTEM WITH MICROPROCESSOR-BASED PROTECTIVE RELAY AND FIBER-OPTIC COMMUNICATION LINK.
In: CIRED . 9th International Conference on Electricity Distribution. Brighton, 1987.
London: IEE, 1989. p. c.11/1-6.

[10] Borlase, S.H.
ADVANCING TO TRUE STATION AND DISTRIBUTION SYSTEM INTEGRATION IN ELECTRIC UTILITIES.
IEEE transactions in power delivery, Vol. 13 (1998), Iss. 1, p. 129-134.

[11] Carrapatoso, A.; F. Gomes
HIGH PERFORMANCE TELECONTROL REMOTE UNIT.
In: ISIE, Proceedings of the 1997 IEEE International Symposium on Industrial Electronics. Guimaraes, 7-11 July 1997.
Piscataway: IEEE, 1997. Vol. 1, No. 48073, p. 99-102.

[12] Corera, J.M. and J.V. Ferrer, J.R. Lezameta, O. Martin, M. Nebreda
IBERDROLA'S ALARMS MANAGEMENT SYSTEM [POWER NETWORK PROTECTION].
In: CIRED. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 7/1-5.

[13] DeZwart, T and J.K. Stoutjesdijk, Y. Fu, H.F.A. Verhaart, G.P. Krikke, C. Kroon, J.W. VanderStoep, W.R Rutgers, F.B.J. Koops
CENTRAM TRANSFORMATORDAG.
Arnhem: KEMA Nederland B.V., 1999.
Contact Information: Centram, P.O. Box 9035, NL-6800 ET, Arnhem, Informant: W. DeLeeuw, Tel.: +31 (0) 26 356 23 23, Fax.: +31 (0) 26 351 36 83, Email: centram@kema.nl.

[14] Dijk, H.E. and F.L. de Rooy
RAAMWERKPLAN INFORMATIE EN COMMUNICATIE TECHNOLOGIE VOOR ENERGIEBEDRIJVEN – CONCEPT.
Arnhem: KEMA Nederland B.V., dept. of KEMA Connect, 1998.
Ordering Information: Report 5007.CON18.P01 (98340007-Connect 98-xxxx).

[15] Feijs, L. and M. Manders
INTERNET CONTROL AND MONITORING - OPTIONS AND COMPLICATIONS -
Eindhoven: Eindhoven University of Technology, Eindhoven Embedded Systems Institute (EESI), June 2000.

[16] Flowers, T. and B. Houle, J. Refzer, R. Ramanathan
ROUTING SCADA DATA THROUGH AN ENTERPRISE WAN.
IEEE Computer Applications in Power, Vol. 8 (1995), Iss. 3, p. 40-4.

[17] Geilen, M.C.W.
BRIEF MANUAL FOR SHESIM, THE MODELLING AND SIMULATION TOOL FOR THE SHE-METHODOLOGY.
Eindhoven: Eindhoven University of Technology, dept. of Electrical Engineering, Information and Communication Systems Group, April 1998.

[18] Eindhoven: Eindhoven University of Technology, dept. of Electrical Engineering, Information and Communication Systems Group, April 1998.Hirsch, M. and M. Johnson, D. Smith
COMMUNICATION ARCHITECTURES FOR CENTRALIZED MONITORING AND CONTROL OF POWER EQUIPMENT.
In: INTELEC. Eighteenth International Telecommunications Energy Conference. Boston, 6-10 Oct. 1996.
New York: IEEE, 1996. Vol. 1, p. 76-81.

[19] Humphreys, S.
SUBSTATION AUTOMATION SYSTEMS IN REVIEW.
IEEE Computer Applications in Power, Vol. 11 (1998), Iss. 2, p. 24-30.

[20] Kirkman, J.M.
NEW DEVELOPMENTS IN DISTRIBUTION SUBSTATION AUTOMATION.
In: Fourth International Conference on 'Trends in Distribution Switchgear'. London, 7-9 Nov. 1994.
London: IEE, 1994. No. 400, p. 44-51.

[21] Koch, W.
INTEGRATED SUBSTATION LOOKS LIKE ONE RTU TO DISPATCHERS.
Electrical World, Vol. 209 (1995), Iss. 12, p. 33-5.

[22] Lehtonen, M.
FAULT MANAGEMENT IN ELECTRICAL DISTRIBUTION SYSTEMS
Espoo: CIRED, Working Group WG03 Fault Management, 22 December 1998.

[23] Lehtonen, M. and E. Antila, M. Seppanen
AN INTEGRATED SOLUTION FOR PROTECTION AND AUTOMATION OF POWER SYSTEMS.
In: CIRED. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 24/1-5.

[24] Leibfried, T.
ONLINE MONITORS KEEP TRANSFORMERS IN SERVICE.
IEEE Computer Applications in Power, Vol. 11 (1998), Iss. 3, p. 36-42.

[25] Lorito, F. and M. Mariani, F. Viaro
ADVANCED DECENTRALISED CONTROL SYSTEM FOR MV/LV DISTRIBUTION SYSTEM.
In: CIRED 97. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 11/1-5.

[26] Lundqvist, B. and H. Kronander, A.J. Mackrell
THE INTEGRATION OF PROTECTION, MONITORING, CONTROL AND COMMUNICATION FUNCTIONS IN MODERN ELECTRICAL HV INSTALLATIONS.
In: Sixth International Conference on Developments in Power System Protection. Nottingham, 25-27 March 1997.
London: IEE, 1997. No. 434, p. 176-9.

[27] Miyatake, K. and H. Yamashita, K. Kitbwaki, H. Suenaga
MULTIPLEX REMOTE INFORMATION EQUIPMENT [POWER SYSTEM MONITORING].
In: INTELEC 97. 19th International Telecommunications Energy Conference. Melbourne, 19-23 Oct. 1997.

New York: IEEE, 1997. Vol. 1, p. 154-8.

[28] Olson, G.L.
ESTABLISHING A REMOTE MONITORING SYSTEM TO ENHANCE POWER SYSTEM RELIABILITY.
In: INTELEC. Eighteenth International Telecommunications Energy Conference. Boston, 6-10 Oct. 1996.
New York: IEEE, 1996. Vol.1, p. 94-8.

[29] Otto, E.
SYSTEEM-GEÖRIËNTEERDE BEVEILIGING – BEVEILIGING VAN MIDDENSPANNINGSNETTEN OP BASIS VAN OVERZICHT.
Vlissingen: Hogeschool Zeeland, Thesis, 1999.

[30] Pinto De Sa, J. and M. Conde
PERFORMANCE ISSUES IN INTEGRATED CONTROL AND PROTECTION SYSTEMS FOR SUBSTATIONS.
In: CIRED. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 17/1-5.

[31] Pozzuoli, M.P.
TRANSMISSION – UNIVERSAL RELAY STREAMLINES SUBSTATION AUTOMATION – PROTECTIVE RELAYS ARE BECOMING LIKE PCS.
Electrical World, Vol. 212 (1998), Issue. 12, p. 47-49.

[32] Ramsey, B.
TECHNICAL OVERVIEW: IPV6: PREPARING FOR THE NEXT GENERATION INTERNET PROTOCOL.
Sunnyvale: Network Products Integrated Systems, Inc., 2000.

[33] Rijanto, prof. dr.-ing. H. and ir. W.F.J. Kersten
ELEKTRISCHE ENERGIETECHNIEK 1.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, 1997.
EUT Report 5799.

[34] Rumbaugh, J. and M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
OBJECT ORIENTED MODELING AND DESIGN.
Englewood Cliffs : Prentice-Hall International, 1991.

[35] Schäffler, T. and C. Kern
MODERNE KOMMUNIKATION IN DER FERNWIRK- UND STATIONSLEITTECHNIK
Elektrizitätswirtschaft, Vol. 96 (1997), Iss. 5, p. 182-6.

[36] Schiemann, H.A.
IP INTERGRATION IN THE UTILITIES ENVIRONMENT
In: Utility Automation Europe, autumn 1999
Pennwell: Kema Consulting, Germany

[37] Shephard, B. and J.C. Tobias, A. Cochet-Muchy
EXPERIENCE ON THE SERIAL INTERFACING OF PROTECTION RELAYS TO TELECONTROL SYSTEMS.
In: Sixth International Conference on Developments in Power System Protection. Nottingham, 25-27 March 1997.

London: IEE, 1997. No. 434, p. 313-18.

[38] Smith, H. Lee
SUBSTATION AUTOMATION PROBLEMS AND POSSIBILITIES.
IEEE Computer Applications in Power, Vol. 9 (1996), Iss. 4, p. 33-36.

[39] Stelian, G. and B. Florin, E. Traian, B. Iacob
DEVELOPMENT OF RENEL SIBIU'S SCADA & MONITORING SYSTEM AND IMPROVEMENT OF PROTECTIVE SYSTEM HANDLING DIGITAL PROTECTION INTEGRATED INTO SCADA SYSTEMS.
In: CIRED. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 9/1-5.

[40] Swartz, L.
INTEROPERABILITY OF INTELLIGENT ELECTRONIC DEVICES IN A SUBSTATION.
In: Fourth International Conference on Power System Control and Management. London, 16-18 April 1996.
London: IEE, 1996. No. 421, p. 187-90.

[41] Tanenbaum, A.S.
COMPUTERNETWERKEN.
Schoonhoven: Academic Service, 1998. Translated by L. Geurts, 2nd edition, 2nd printing.

[42] Tapper, M.
BUDGET APPROACH TO SUBSTATION AUTOMATION.
In: Proceedings of the 1996 IEE Colloquium on Methods of Substation Automation. London, 1996.
Stevenage: IEE, 1996. No. 163, p. 4/1-5.

[43] Thang, W.Y. and J.Y. Bouission, B. Peruzzo, R. Hubner
AN APPROACH FOR AN OPEN CONTROL SYSTEM FOR SUBSTATIONS.
In: CIRED 97. 14th International Conference and Exhibition on Electricity Distribution. Part 1: Contributions. Birmingham, 2-5 June 1997.
London: IEE, 1997. No. 438, Vol. 4, p. 8/1-5.

[44] Vahamaki, O.J. and A.J. Allen, J.T. Gaff
HIGH SPEED PEER-TO-PEER COMMUNICATION SYSTEM FOR INTEGRATED PROTECTION AND CONTROL IN DISTRIBUTION NETWORKS.
In: Sixth International Conference on Developments in Power System Protection. Nottingham, 25-27 March 1997.
London: IEE, 1997. No. 434, p. 243-6.

[45] VandenHeuvel, W.M.C.
GRONDSLAGEN VAN DE ELEKTRISCHE ENERGIETECHNIEK.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, 1994.
EUT Report 5740, 3rd edition.

[46] VanderPutten, P.H.A.
SYSTEM LEVEL DESIGN METHODOLOGY
http://www.ics.ele.tue.nl/~piet/she-presentation/May98-ppt97/index.htm

[47] VanderPutten, P.H.A. and J.P.M. Voeten, M. Geilen

SPECIFICATION OF COMPLEX HARDWARE/SOFTWARE SYSTEMS - SOFTWARE/HARDWARE ENGINEERING (SHE) -
Eindhoven: Eindhoven University of Technology, dept. of Electrical Engineering, Information and Communication Systems Group, 1998.

[48] VanKruining, Ir. P.C.M. and Ir. J.H.P. Lommert, Prof. ir. H.H. Overbeek, Ir. R.J.R. Waumans
ELEKTRICITEITSDISTRIBUTIENETTEN.
Arnhem: EnergieNed, 1st edition, 1996.
Deventer: Kluwer Bedrijfsinformatie b.v., 1996.

[49] Verhoeven, B.
DIAGNOSTIC VIEW ON MAINTENANCE FOR UTILITIES
Arnhem: KEMA Nederland B.V., Dept. of KEMA TDP, August 1999.
Internal Presentation – filename: "Bas Verh. aug. 1999 Diagnostic view on maintenance.ppt" (ask B.J.M Janssen)

[50] Verhoeven, B.
TECHNIQUES AND EQUIPMENT FOR MONITORING
Arnhem: KEMA Nederland B.V., Dept. of KEMA TDP, August 1999.
Internal Presentation – filename: "Bas Verh. aug. 1999 Techniques and Equipment for Monitoring.ppt" (ask B.J.M Janssen)

[51] Vermaas, C.P.T. and I. Biesmond, E. VandeBurgwal
INTEGRATIEPROJECT VOOR VKO – ENERGIEVOORZIENING MEERHOVEN.
Eindhoven: Eindhoven University of Technology, Faculty of Technology Management (TeMa), Internal project report, 1999.

[52] Wakkerman, L.J.C.
TUSSENRAPPORT OVER FAALOORZAKEN VAN DISTRIBUTIE- EN KOPPELTRANSFORMATOREN OP BASIS VAN DE ONVOORZIENE NIET-BESCHIKBAARHEID".
Arnhem: KEMA Nederland B.V., dept. of KEMA TDP, 13 September1994.
Ordering information: Report 43613-T&D 94-102278.

[53] (Standard)
IEEE GUIDE FOR DEVELOPING SYSTEM REQUIREMENTS SPECIFICATIONS (STANDARD 1233-1996)
New York: IEEE, 1996.

[54] (Standard)
IEEE RECOMMENDED PRACTICE FOR SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) (STANDARD 830-1993)
New York: IEEE, 1994.

Epilogue

At the end of this report, I will mention some of my experiences during the project.

My initial expectations were that the project would focus on detailed data communication issues, like an investigation on the usability of protocols and analysis of existing networks. However, the project turned out to be much more complex than expected, which required a higher level of approach. Before the design of a communication network, a specification needs to be made-up, as a result of analyzing the subject's application domain. This made my primary goal change towards demonstrating a possible approach of developing these kinds of complex systems. Modeling has played a central role throughout the process of specification of a system.

In the beginning, my experiences with Object Oriented techniques were marginal. However, after a while the object-oriented philosophy became more and more clear to me: Data and behavior should be combined in object classes, instead of the traditional way of thinking that data and behavior are to be treated separately. The actual modeling in SHE made these concepts much more clear. I experienced that this approach is very important to the process of analyzing the system, because during this phase, lots of questions arose about various aspects in the problem domain.

Therefore, close communication between energy and information engineering is very important. It would have been better if a dedicated project team existed or had been created, in which the author participated than to assign a multidisciplinary project to a 'single' person.

As a conclusion can be said that I learned a lot about the design complexity of large (communication) systems and the approach to deal with it.

Appendix A: **System Requirement Specification (SyRS)**

i Table of Contents

ii List of Figures

iii List of Tables

1. Introduction

1.1 System Purpose
a) Purpose
b) Intended audience

1.2 System Scope
a) Identify system by name
b) Results of earlier analysis; what will and won't the product do?
c) System application description (top-level benefits, objectives and goals)

1.3 Definitions, Acronyms and Abbreviations

1.4 References
a) Complete list of all documents
b) Title, report number, date, publishing organization
c) Sources

1.5 System Overview
a) SyRS contents
b) SyRS organization

2. General System Description

2.1 System Context
(Diagrams and narrative)
a) Overview of the context of the system
b) All significant interfaces crossing the system's boundaries

2.2 System Modes and States

2.3 Major System Capabilities

2.4 Major System Conditions

2.5 Major System Constraints
a) Regulatory policies
b) Hardware limitations (e.g. timing)
c) Interfaces to other applications
d) Parallel operation
e) Audit functions
f) Control functions
h) Signal handshake protocols (Xon/Xoff, Ack/Nak)
i) Reliability requirements
j) Criticality of the application
k) Safety and security considerations

2.6 User Characteristics

(Identify each type of user (function, location, type of device))

2.7 Assumptions and Dependencies
(Factors that affect the requirements stated in SyRS)

2.8 Operational Scenarios
(Descriptive examples of how the system will be used)

3. System Capabilities, Conditions and Constraints
(Note – System Behavior, Exception Handling, Manufacturability and Deployment should be covered under each Capability, Condition and Constraint.)

3.1 Physical
3.1.1 Construction
(Environmental characteristics of where the system will be installed)
3.1.2 Durability
3.1.3 Adaptability
(Growth, expansion, capability, contraction)
3.1.4 Environmental Conditions
(natural, induced and electromagnetic signal environment)

3.2 System Performance Characteristics
a) Dynamic actions or changes that occur
b) Endurance capabilities
(e.g. minimum total life expectancy, required operational session duration, planned utilization rate)
c) Performance requirements for the operational phases and modes

3.3 System Security
a) Facility that houses the system
b) Operational security requirements

3.4 Information Management

3.5 System Operations
3.5.1 System Human Factors
(Areas, stations or equipment that would require concentrated human engineering attention, due to the sensitivity of the operation or criticality of the task)
3.5.2 System Maintainability
(Time, rate, maintenance complexity, maintenance action indices)
3.5.3 System Reliability
(In quantitative terms, to be met under which conditions)

3.6 Policy and Regulation
(Organizational policies, external regulatory requirements/constraints)
a) Health and safety criteria
b) Toxic systems and electromagnetic radiation

3.7 System Life Cycle Sustainment
(Quality activities: review, measurement collection and analysis)

4. System Interfaces
(Interfaces among different components and their external capabilities, including all its users, both human and other systems)

Appendix B **Sensor specialization tree**

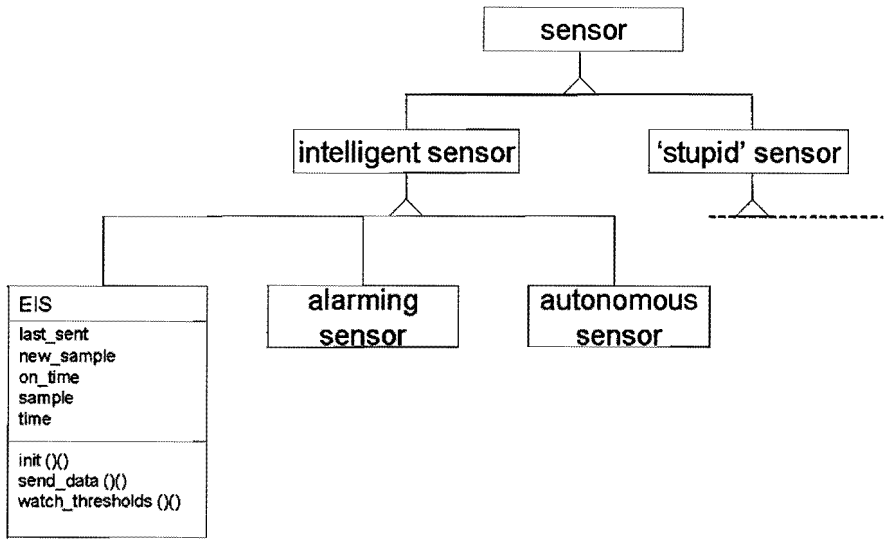


Figure 7.1 Sensor specialization tree

Figure 7.1 shows an Object class diagram of sensors, resulting from an earlier short exploration in the application domain. It gives an impression of the possible subclasses of superclass 'sensor', and their mutual relationships.
A sensor generally transforms a physical quantity into measurable signals.

Sensors can be classified to its intelligence. 'Stupid' sensors can be seen as the classical passive ones. They only carry out measurements. It is the user's responsibility to process the sensor's signals. There is no communication between sensor and user.
Intelligent sensors, conversely, have behavior, that allows them to communicate with the user. The intelligence makes it possible to respond on requests of the user to carry out a certain task. The internal behavior takes care of those tasks after which possible results are communicated in one way or the other. An example is when a sensor is ordered to directly communicate the most recent measurement results.

Three variants of intelligent sensors are the autonomous sensor, the alarming sensor and the Enhanced Intelligent Sensor (EIS).
An *autonomous sensor* has the capability of periodically sending messages containing measurement results. The timing functionality is embedded into the sensor and the interval is adjustable by the user. The sensor's task could be "communicate measurement results every 10 minutes".
An *alarming sensor* has the capability to process the sensor's signal. An example is the comparison of measured data to certain thresholds. As soon as the value of the measured quantity exceeds the threshold, the sensor signals the user. The threshold values may be adjusted by the user, of course. The output of the sensor becomes dependent on the measurement results and thereby event driven.
The third type of sensor, the *Enhanced Intelligent Sensor (EIS)*, is capable of performing both tasks mentioned above.

The sensor used in the example model (T-oil) is a specialization of the class 'EIS' and represents the value of the oil temperature. Although at this moment solely the 'general' intelligent behavior is used (responding to requests for recent measurement results), both additional features are to be used in the future.

Appendix C: **POOSL Specification file '*MICS_final_TL.sim*'**

```
drawing
system

instances
process of class 'CEMS' in 48@24 to 184@96 name 'CEMS' at 96@40
connectors
'substations' at 112@96 name at 80@64

scenarios
'normal'
multiple cluster of class 'substation' number: 2 in 40@160 to 192@256 name 'EHV_substation' at 64@208
connectors
'CEMS' at 112@160 name at 96@168

scenarios
'normal'

channels
name 'energy_management' at 120@104 message at 120@119 margin 673
nodes

segments
segment from socket 'substations' of instance 'CEMS' to socket 'CEMS' of instance 'EHV_substation'

scenarios
'normal'


scenarios
'normal', 'all'

poosl specification

process class FRA(analyser_ID: String)
/* no superclass */

instance variables
changing: Boolean; reserved: Boolean; in_use: Boolean; target_reserved: String; target_in_use: String; under_maintenance: Boolean

communication channels
TMA, TMA_error

message interface
TMA ! NOT_available(UNKNOWN, UNKNOWN);
TMA_error ! error_procedure(UNKNOWN);
TMA ! available(UNKNOWN, UNKNOWN, UNKNOWN);
TMA ? conditions_OK(STRING);
TMA ! go_back_online(UNKNOWN, UNKNOWN);
TMA ? online_again(STRING);
TMA ? conditions_failed(STRING, STRING);
TMA_error ? abort_now(STRING, STRING);
TMA ? analyser_request(STRING);
TMA_error ? something_wrong(STRING, STRING, STRING)

initial method call
init()()

instance methods
init ()()
/*==============================
    instance variables explanation
==============================
analyzer_ID:                    the identity of this analyzer
changing:                       to prevent local variables from being changed by two parallel methods
                                (may be replaced by the use of "atomic" operator {...} later on)
in_use:                         indicates if the analyzer is now being used
reserved:                       the analyzer is in_use AND a new component claimed the availability afterwards
target_in_use:                  component attached to the analyzer
target_reserved:                component that will be attached to the analyzer after finishing the current analysis
under_maintenance:              indicates that the analyzer is unavailable, due to maintenance */

/*==============================
    local variables declaration
==============================*/

/*==============================
    (local) variables initialization
==============================*/
changing:= false;               /* changes are allowed */
in_use:= false;                 /* in the beginning not yet in use... */
reserved:= false;               /* in the beginning not yet reserved */
under_maintenance:= false;      /* in the beginning not yet under maintenance */
```

```
/*================================
    POOSL behaviour description
================================*/
par                                 /* parallel execution op three methods */
    watch_requests()()              /* wait for FRA requests */
and
    watch_condition_results()()     /* wait for the results of the currently attached component */
and
    watch_errors()()                /* wait for error messages (and handle them) */
rap.

start_analysis ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/
IFRA_timer: INTEGER;                /* time before next FRA (can be modified in between, depending on analysis results) */
random_var: RANDOMGENERATOR;        /* used for calculation of FRA_timer */
target: STRINGI                     /* identity of the currently attached component (for use of conditionally receiving messages)
*/


/*================================
    local variables initialization
================================*/
random_var:= new(RandomGenerator); /* gives FRA_timer a more variable character (depending on the previous FRA results) */

/*================================
    POOSL behaviour description
================================*/
delay 173;                          /* process time */
[changing not]changing:=true;       /* make sure no other method is changing instance variables "in_use", "reserved", "target_in
                                    use", or "target_reserverd" */
/* FRA_timer:= 25000 + (175000 * (random_var randomizeSeed random)) asInteger; */
                                    /* these values are realistic, but do not cause a second offline request shortly after another
                                    one */
FRA_timer:= 3500 + (1750 * (random_var randomizeSeed random)) asInteger;
                                    /* this will go wrong more often (for demonstration purposes) */
TMA!go_back_online(target_in_use, FRA_timer);
                                    /* tell TMA to go online again, after finishing measurements */
TMA?online_again(target!target=target_in_use);
                                    /* wait for confirmation */
in_use:=reserved;                   /* pass on reserved status to in_use status */
target_in_use:=target_reserved;     /* pass on the target's identity for next FRA */
if   reserved                       /* if there was a reservation already... */
then
    TMA!available(target_reserved, analyser_ID, "offline, T_oil < 308K");
                                    /* ...the reserved target must be sent a "go" message */
fi;
reserved:= false;                   /* reset variable */
target_reserved:= "";               /* reset variable */
changing:=false.                    /* end of changing -> changes allowed again */

watch_errors ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/
ladvise: STRING;                    /* advise in the direction of a solution */
error_source: STRING;               /* indicates the source of the problem */
problems: STRING;                   /* indicates the kind of problem */
target: STRINGI                     /* identity of component of which errors are expected */

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/
sel
    TMA_error?something_wrong(target, error_source, problems);
                                    /* in this model: expecting a problem on FRA channel */
    sel
        advise:="try again"         /* problem probably solved by now: try again */
    or
        advise:="abort"             /* advise TMA to abort its process */
    les;
    TMA_error!error_procedure(advise); /* start error procedure at TMA and follow advise */
or
```

```
    TMA_error? abort_now(target, problems)(target=target_in_use) not);
                                        /* direct message and order to abort the process */
    abort_process(target, problems)()   /* abort FRA of target */
les;
watch_errors ()().                      /* wait for new requests -> infinite behavior */

start_process ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/

/* to be modeled in the future: */
/* preparation, etc. */

delay 28;                               /* process time */
start_measuring()().                    /* proceed to starting measurements */

start_measuring ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/

/* to be modeled in the future: */
/* measuring process()() */

delay 59;                               /* process time */
start_analysis()().                     /* proceed to results analysis */

watch_condition_results ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/
lcause: STRING;                         /* used for the cause of failing conditions */
target: STRING!                         /* identity of component of which condition results are expected */

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/
sel
    TMA?conditions_OK(target!target=target_in_use);
                                        /* receive positive condition results from expected component */
    start_process()()                   /* if conditions OK, then start measuring */
    abort   (                           /* process may be aborted by the reception of an error message */
            TMA_error?abort_now(target, cause!target=target_in_use);
                                        /* in this case an order to abort the process directly */
            abort_process(target, cause)();
                                        /* obey orders */
            );
or
    TMA?conditions_failed(target, cause!target=target_in_use);
                                        /* receive negative condition results from expected component */
    abort_process(target, cause)();     /* in this case, the process should be aborted as well */
les;
watch_condition_results ()().           /* wait for new condition results -> infinite behavior */
```

```
watch_requests ()()
/*================================
    method parameters explanation
================================*/

/*================================
    local variables declaration
================================*/
|target: STRING|                          /* identity of component */

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/
TMA?analyser_request(target);            /* reception of a FRA request */
delay 2;                                 /* process time */
if   reserved|under_maintenance          /* if not available */
then
    TMA!NOT_available(target, analyser_ID) abort delay 30
                                         /* try and send "not_available" to TMA for 30 minutes */
                                         /* if no rendezvous then just proceed */
else                                     /* if available */
    [changing not]changing:=true;        /* make sure no other method is changing instance variables "in_use", "reserved", "target_in
                                         use", or "target_reserverd" */
    sel
        [(in_use not)&(under_maintenance not)] TMA!available(target, analyser_ID, "offline, T_oil < 308K");
                                         /* inform target about availability and the conditions to be met before FRA can proceed */
        target_in_use:= target;          /* attach component's identity */
        in_use:= true                    /* set variable */
    or
        [(in_use)&(reserved not)&(under_maintenance not)] TMA!available(target, analyser_ID, "wait");
                                         /* tell target to wait for conditions (meaning that a reservation has been made for this
                                         component */
        target_reserved:= target;        /* make reservation */
        reserved:= true                  /* set variable */
    or
        delay 30                         /* watchdog of 30 minutes */
    les;
    changing:=false;                     /* end of changing -> changes allowed again */
fi;
watch_requests ()().                     /* wait for new requests -> infinite behavior */

abort_process (target:STRING; problems: STRING)()
/*================================
    method parameters explanation
================================
problem:                                 reason for aborting
target:                                  identity of component to be aborted (in_use or reserved) */

/*================================
    local variables declaration
================================*/
|FRA_timer: INTEGER|                     /* variable that holds the time before a new FRA should be carried out */

/*================================
    local variables initialization
================================*/
FRA_timer:=2880;                         /* when aborted: try again in two days */

/*================================
    POOSL behaviour description
================================*/
delay 36;                                /* process time */
                                         /* probably: warn_supervisor()() to be added later */
[changing not]changing:=true;            /* make sure no other method is changing instance variables "in_use", "reserved", "target_in
                                         use", or "target_reserverd" */
if   (target=target_in_use)              /* meaning the FRA of the component currently in use should be aborted */
then
    if   (problems="no_offline_permission_by_CEMS") not
                                         /* the reason for abortion */
    then
                                         /* later: make analyzer available again and prepare component (and personnel) to go online
                                         again */
        TMA!go_back_online(target_in_use, FRA_timer);
                                         /* tell TMA to go online again */
        TMA?online_again(target|target=target_in_use);
                                         /* wait for confirmation */
    fi;
    in_use:=reserved;                    /* pass on reserved status to in_use status */
    target_in_use:=target_reserved;      /* pass on the target's identity for next FRA */
    if   reserved                        /* if there was a reservation already... */
    then
```

```
        TMA!available(target_reserved, analyser_ID, "offline, T_oil < 308K");
                                        /* ...the reserved target must be sent a "go" message */
    fi;
    reserved:= false;                   /* reset variable */
    target_reserved:= ""                /* reset variable */
else
    if   target=target_reserved         /* in reserved target must be aborted... */
    then
        reserved:= false;               /* ...then just reset variable */
        target_reserved:= "";           /* reset variable */
    fi;
fi;
changing:=false.                        /* end of changing -> changes allowed again */



process class TMA(component_ID: String)
/* no superclass */

instance variables
estimated_FRA_duration: Integer; FRA_timer: Integer

communication channels
analyser_error, analyser, TEA, TEMP

message interface
analyser_error ? error_procedure(STRING);
TEMP ? T_oil(BOOLEAN, STRING);
analyser ? NOT_available(STRING, STRING);
analyser_error ! something_wrong(UNKNOWN, UNKNOWN, UNKNOWN);
analyser ? go_back_online(STRING, UNKNOWN);
TEMP ! check_oil_temperature(UNKNOWN, UNKNOWN);
TEA ? online_status(BOOLEAN, STRING);
analyser ! conditions_OK(UNKNOWN);
TEA ? offline_status(BOOLEAN, STRING);
analyser ! online_again(UNKNOWN);
analyser_error ! abort_now(UNKNOWN, UNKNOWN);
analyser ! conditions_failed(UNKNOWN, UNKNOWN);
analyser ! analyser_request(UNKNOWN);
TEA ! go_offline(UNKNOWN, UNKNOWN);
analyser ? available(STRING, STRING, STRING);
TEA ! go_online()

initial method call
init()()

instance methods
watch_schedule ()()
/*================================
    method parameters explanation
================================
*/

/*================================
    local variables declaration
================================*/
lanalyser_available: BOOLEAN;       /* true if the analyzer is available */
analyser_ID: STRING;                /* the identity of the analyzer attached to the maintenance process (FRA) */
conditions: STRING;                 /* conditions for performing an FRA (determined by FRA) */
watchdog_conditions: INTEGERI       /* maximum duration to meet these conditions */

/*================================
    local variables initialization
================================*/
watchdog_conditions:= 2880;         /* two days at maximum */

/*================================
    POOSL behaviour description
================================*/
delay FRA_timer;                    /* time for a new FRA for this component when delay finished */
request_for_analysis ()(analyser_available, analyser_ID, conditions);
                                    /* method claims (asks for) the availability of FRA */
if   analyser_available
then
    handle_conditions(analyser_ID, conditions, watchdog_conditions)();
                                    /* meet the conditions determined by analyzer */
else
    FRA_timer:= 1440;               /* if analyzer is not available and FRA cannot be scheduled, means sending a new request in
                                    1 day */
fi;
watch_schedule()().                 /* wait for new request: infinite behavior */

init ()()
/*================================
    instance variables explanation
```

```
================================
component_ID:                    the identity of this component, e.g. characterized by the following parameters (implicitly or
                                 explicitly):
                                     kind of transformer, serial number, number of the standard applicable, number of phases
                                     manufacturer's name, year of manufacturing, type of cooling, connection symbol
                                     rated power, rated frequency, rated voltages, rated currents, short circuit impedance
                                     mass of insulating oil, total mass
estimated_FRA_duration:          estimated offline duration required by the FRA process
FRA_timer:                       goes off when the time has come for a new FRA */

/*==============================
    local variables declaration
==============================*/
|random_var: RANDOMGENERATOR|   /* used for calculation of FRA_timer */

/*==============================
    (local) variables initialization
==============================*/
random_var:= new(RandomGenerator); /* gives FRA_timer a more variable character (depending on the previous FRA results) */
estimated_FRA_duration:= 268;    /* later on: this variable can be calculated on the basis of a/o. previous FRA results */
/* FRA_timer:= 25000 + (175000 * (random_var randomizeSeed random)) asInteger; */
                                 /* these values are realistic, but do not cause a second offline request shortly after another
                                 one */
FRA_timer:= 3500 + (1750 * (random_var randomizeSeed random)) asInteger;
                                 /* this will go wrong more often (for demonstration purposes) */

/*==============================
    POOSL behaviour description
==============================*/
watch_schedule()().              /* wait for next scheduled action */

handle_conditions(analyser_ID: STRING; conditions: STRING; watchdog_conditions: INTEGER)()
/*==============================
    method parameters explanation
==============================
analyser_ID:                     the identity of the analyzer attached to the maintenance process (FRA)
conditions:                      conditions for performing an FRA (determined by FRA)
watchdog_conditions:             maximum duration to meet these conditions */

/*==============================
    local variables declaration
==============================*/
|max_T_oil: INTEGER;             /* will be received from the request's originator: "let me know when T_oil is below this value"
*/
offline_status: BOOLEAN;         /* this variable indicates if this component successfully went offline */
offline_problems: STRING;        /* indicates the reason for not going offline, acquired via a message from CEMS */
online_status: BOOLEAN;          /* this variable indicates if this component is online again */
online_problems: STRING;         /* indicates the reason for not going online, acquired via a message from CEMS */
target: STRING;                  /* conditional message reception: checks if FRA"s response is addressed to this component
                                 */
temperature_problems: STRING;    /* e.g. "temperature timeout" */
temperature_status: BOOLEAN;     /* true if T_oil meets the requirement (below max_T_oil) */
trash: STRING|                   /* used for parameters from received messages with that will not be used (only clarifying the
                                 message) */

/*==============================
    local variables initialization
==============================*/
max_T_oil:= 0;                   /* self-explanatory */
offline_problems:= "";           /* self-explanatory */
offline_status:= false;          /* self-explanatory */
online_status:= true;            /* self-explanatory */
temperature_problems:= "";       /* self-explanatory */
temperature_status:= false;      /* self-explanatory */

/*==============================
    POOSL behaviour description
==============================*/
delay 6;                         /* process_time */
if   conditions="wait"           /* apparently, the FRA was occupied, and the request has been scheduled */
then
    analyser?available(target, analyser_ID, conditions | target=component_ID)
                                 /* wait for a message from FRA, confirming its availability */
    abort delay 2600             /* if this message does not show up in the next (almost) two days, then abort */
fi;
if   (conditions="offline, T_oil < 308K")  /* the FRA requires the component to be offline with T_oil = 308K at maximum */
then
    max_T_oil:=308;              /* for safety reasons (e.g. for personnel), a maximum oil temperature of 308K is determined
                                 */
                                 /* at this moment assuming the transformer is still online */
                                 /* later on, this can be checked */
    (                            /* handling first condition (offline) */
    TEA!go_offline(analyser_ID, estimated_FRA_duration);
                                 /* "analyser_ID" will be sent along as a parameter, because the CEMS uses it for determing
                                 priority */
        TEA?offline_status(offline_status, offline_problems);
```

```
                                        /* wait for message that indicates if the offline status is reached */
        if   offline_status not         /* if the offline status has not been reached */
        then
            error_handling("FRA_conditions", offline_problems, 0)(trash)
                                        /* handle cause of problems as a parameter */
                                        /* maybe a number of attempts van be added later */
        else                            /* handling second condition (temperature) */
            TEMP!check_oil_temperature("Max_temp: ", max_T_oil);
                                        /* ask for T_oil response, when condition is met */
            TEMP?T_oil (temperature_status, temperature_problems);
            if   temperature_status not
            then
                error_handling("FRA_conditions", temperature_problems, 0)(trash)
                                        /* handle cause of problems as a parameter */
                                        /* maybe a number of attempts van be added later */
            fi;
        fi)

/*          dit bericht moet komen voordat de timer afloopt. Het TEMPproces kan de temperatuurdaling extrapoleren en
            een schatting maken van de tijd van "crossing threshold". Ligt deze beduidend later dan de maximum tijd
            gegevens in de parameter van het bericht, dan kan er een waarschuwingsbericht volgen, met als parameter
            het verwachte tijdstip van afkoeling. De maintenance_agent kan vervolgens besluiten hierop in te gaan en te
            wachten, maar kan ook besluiten te aborteren. (misschien via de FRA?!?)
*/

    abort
        (delay watchdog_conditions;     /* aborted by the watchdog if condition handling takes too long */
        error_handling("FRA_conditions", "timeout", 0)(trash)
                                        /* followed by error handling */
        );
fi;
delay 4;                                /* process time */
if   offline_status
then
    if   temperature_status             /* if all conditions are met */
    then
        analyser!conditions_OK (component_ID);
                                        /* notify the FRA of being ready for the measurement */
    fi;
    analyser?go_back_online (target, FRA_timer|target=component_ID);
                                        /* wait for message after finishing measurements */
                                        /* if temperature condition is not met, this message follows as well, */
                                        /* because component must go back online as quickly as possible */
    TEA!go_online ();                   /* communicate to Energy Agent */
    TEA?online_status (online_status, online_problems);
                                        /* wait for confirmation, or possible problems */
    analyser!online_again(component_ID)
                                        /* inform FRA, so that is will become available to other components again */
fi.

error_handling(error_source: STRING; problems: STRING; attempts: INTEGER)(attempts: INTEGER)
/*==============================
    method parameters explanation
==============================
attempts:                               number of times the process tried and waited for an action
error_source:                           indicates the source of the problem
problems:                               indicates the kind of problem */

/*==============================
    local variables declaration
==============================*/
|max_attempts_request: INTEGER;        /* self-explanatory */
max_attempts_response: INTEGER|       /* self-explanatory */

/*==============================
    local variables initialization
==============================*/
max_attempts_request:= 3;               /* self-explanatory */
max_attempts_response:= 3;              /* self-explanatory */

/*==============================
    POOSL behaviour description
==============================*/
attempts:=attempts+1;                   /* adjust the number of attempts */
sel                                     /* at this moment 3 possible options */
    [(error_source="FRA_channel")&(problems="request timed out")]
    if   attempts>max_attempts_request
    then
        analyser_error!something_wrong(component_ID, error_source, problems)
                                        /* if maximum attempts exceeded: send message to analyzer via error-channel */
                                        /* if not: this method ends and returns a new (higher) number of attempts */
                                        /* analyzer will decide how to handle */
    fi;
or
    [(error_source="FRA_channel")&(problems="response timed out")]
    if   attempts>max_attempts_response
```

```
    then
        analyser_error!something_wrong(component_ID, error_source, problems)
                                        /* if maximum attempts exceeded: send message to analyzer via error-channel */
                                        /* if not: this method ends and returns a new (higher) number of attempts */
                                        /* analyzer will decide how to handle */
    fi;
or
    [error_source="FRA_conditions"]
    analyser!conditions_failed(component_ID, problems)
                                        /* inform the analyzer about the problem. the analyser will send a "back online" message */
les;
delay 3.                                /* process time */

request_for_analysis ()(available: BOOLEAN; analyser_ID: STRING; conditions: STRING)
/*==============================
    method parameters explanation
==============================
analyser_ID:                            the identity of the analyzer attached to the maintenance process (FRA)
available:                              true if the analyzer is available
conditions:                             conditions for performing an FRA (determined by FRA) */

/*==============================
    local variables declaration
==============================*/
ladvise: STRING;                        /* after a number of attempts, error handling will advise whether to try again or to abort */
attempts_to_receive: INTEGER;           /* self-explanatory */
attempts_to_send: INTEGER;              /* self-explanatory */
failure: BOOLEAN;                       /* "true" if an attempt to send or receive a message within a certain time failed */
received: BOOLEAN;                      /* "true" when successfully received */
sent: BOOLEAN;                          /* "true" when successfully sent */
target: STRING;                         /* conditional message reception: checks if FRA"s response is addressed to this component
                                        */
watchdog_request: INTEGER;              /* maximum time to try and send an FRA request, before error handling is called */
watchdog_response: INTEGERI             /* maximum time to try and receive an FRA response, before error handling is called */

/*==============================
    local variables initialization
==============================*/
attempts_to_receive:= 0;                /* self-explanatory */
attempts_to_send:= 0;                   /* self-explanatory */
failure:= false;                        /* self-explanatory */
received:= false;                       /* self-explanatory */
sent:= false;                           /* self-explanatory */
watchdog_request:= 10 ;                 /* maximum time to wait before request is communicated is 10 minutes */
watchdog_response:=7;                   /* maximum time to wait before response is communicated is 7 minutes */

/*==============================
    POOSL behaviour description
==============================*/
while   (sent not)&(failure not)        /* if a failure has been reported, proceed with next while-do loop */
do
    sel                                 /* a choice should be made: error-message reception or successfully send the request
                                        message */
        analyser_error?error_procedure(advise);
                                        /* possible error messages will be treated first */
        if   advise="try again"         /* after a number of times trying to send a request, try again for n times */
        then
            attempts_to_send:=0;        /* start counting from 0 */
        else
            failure:=true               /* if not "try again" then a failure is reported */
        fi;
    or
        delay 1;                        /* error messages should be the first choice (instead of non-deterministic choice) */
        analyser!analyser_request (component_ID);
                                        /* try and sent a request to the analyzer */
        sent:= true;                    /* if succeeded: sent = true */
    or
        delay watchdog_request;         /* if request timed out... */
        error_handling("FRA_channel", "request timed out", attempts_to_send)(attempts_to_send);
                                        /* ...an error handling procedure will follow */
    les;
od;
while   (received not)&(failure not)    /* if a failure has been reported, proceed to the point after this while-do loop */
do
    sel                                 /* a choice should be made: error-message reception or successfully receive a response
                                        message */
        analyser_error?error_procedure(advise);
                                        /* possible error messages will be treated first */
        if   advise="try again"         /* after a number of times trying to receive a response, try again for n times */
        then
            attempts_to_receive:=0;     /* start counting from 0 */
        else
            failure:=true               /* if not "try again" then a failure is reported */
        fi;
    or
        delay 1;                        /* error messages should be the first choice (instead of non-deterministic choice) */
```

```
        analyser?available(target, analyser_ID, conditions I target=component_ID);
                                        /* wait for a positive response from the analyzer */
        received:= true;                    /* if succeeded: received = true */
        available:= true;               /* if succeeded: available = true */
    or
        delay 1;                        /* error messages should be the first choice (instead of non-deterministic choice) */
        analyser?NOT_available(target, analyser_ID I target=component_ID);
                                        /* wait for a positive response from the analyzer */
        received:= true;                /* if succeeded: received = true */
        available:= false;              /* for this response: available = false */
    or
        delay watchdog_response;        /* if response timed out... */
        error_handling("FRA_channel", "response timed out", attempts_to_receive)(attempts_to_receive);
                                        /* ...an error handling procedure will follow */
    les;
od;
if  failure                             /* if there were problems with sending or receiving, there might be something wrong with the
                                        FRA-channel */
then
    available:= false;                  /* this means that the FRA is not available */
    analyser_error!abort_now(component_ID, "FRA_channel communication problems");
                                        /* report error via FRA_error_channel */
fi.



cluster class substation(substation_ID: String)

communication channels
CEMS

message interface
CEMS ? NO_online_permission(STRING, STRING, STRING, INTEGER);
CEMS ! request_online_permission(UNKNOWN, UNKNOWN);
CEMS ! request_offline_permission(UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN);
CEMS ? online_permission(STRING, STRING, STRING, INTEGER);
CEMS ? offline_permission(STRING, STRING, STRING, INTEGER, STRING, INTEGER);
CEMS ? NO_offline_permission(STRING, STRING);
CEMS ! online_confirmation(UNKNOWN, UNKNOWN);
CEMS ! offline_confirmation(UNKNOWN, UNKNOWN)

behaviour specification
(transformer_2: transformer("transformer2", substation_ID)[error_channel/analyser_error, analyser_channel/analyser,
energy_management/CEMS] II Freqency_Response_Analyser: FRA("Frequency Response Analyser")[analyser_channel/TMA,
error_channel/TMA_error] II transformer_1: transformer("transformer1", substation_ID)[error_channel/analyser_error,
analyser_channel/analyser, energy_management/CEMS])\{error_channel, analyser_channel}[CEMS/energy_management]



process class TEA(component_ID: String; substation_ID: String)
/* no superclass */

instance variables
online_status: Boolean

communication channels
TMA, CEMS

message interface
CEMS ? NO_online_permission(STRING, STRING, STRING, INTEGER);
CEMS ! request_online_permission(UNKNOWN, UNKNOWN);
TMA ! online_status(UNKNOWN, UNKNOWN);
TMA ! offline_status(UNKNOWN, UNKNOWN);
CEMS ! request_offline_permission(UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN);
CEMS ? online_permission(STRING, STRING, STRING, INTEGER);
CEMS ? offline_permission(STRING, STRING, STRING, INTEGER, STRING, INTEGER);
TMA ? go_offline(STRING, INTEGER);
CEMS ! online_confirmation(UNKNOWN, UNKNOWN);
CEMS ? NO_offline_permission(STRING, STRING);
TMA ? go_online();
CEMS ! offline_confirmation(UNKNOWN, UNKNOWN)

initial method call
init()()

instance methods
handle_online_requests ()()
/*==============================
    method parameters explanation
==============================
*/
/*==============================
    local variables declaration
==============================*/
lanalyser_ID: STRING;               /* the identity of the analyzer that no longer requires offline status */
```

```
online_problems: STRING;                /* indicates the reason for not going online, acquired via a message from CEMS */
proposed_delay: INTEGER;                /* CEMS suggests a delay before actually going back online */
target_component: STRING;               /* conditional message reception: checks if CEMS"s response is addressed to this
component */
target_substation: STRING;              /* conditional message reception: checks if CEMS"s response is addressed to this substation
*/
trash: STRING;                          /* used for parameters from received messages with that will not be used (only clarifying the
message) */
urgency: INTEGERI                       /* for priority usage */

/*==============================
   local variables initialization
==============================*/

/*==============================
   POOSL behaviour description
==============================*/
TMA?go_online();                        /* waits for online_request_messages from TMA */
CEMS!request_online_permission (component_ID, substation_ID);
                                        /* request must be directed to CEMS */
sel                                     /* either permission or NO_permission can be the CEMS"s response */
   CEMS?online_permission (target_component, target_substation, trash, proposed_delay
                        |((target_component=component_ID)&(target_substation=substation_ID)));
                                        /* conditional reception of the message: check addressing */
                                        /* the indices that are good for the readability of the message are moved to trash */
   delay proposed_delay;                /* follow the advise to wait for some time before actually going back online */
   switch_on ()(online_problems);       /* switch on and report possible problems */
or
   CEMS?NO_online_permission (target_component, target_substation, trash, proposed_delay
            |((target_component=component_ID)&(target_substation=substation_ID)));
                                        /* conditional reception of the message: check addressing */
                                        /* not further implemented in this model */
   online_status:= false;               /* component stays offline */
   online_problems:= "no_online_permission_by_CEMS";
  /* do_something_else()() */           /* no permission: decide when to do another proposition */
les;
TMA!online_status(online_status, online_problems).
                                        /* returns a status message to the request"s originator */

init ()()
/*==============================
   instance variables explanation
==============================
component_ID:                           the identity of this component, e.g. characterized by the following parameters (implicitly or
                                        explicitly):
                                            kind of transformer, serial number, number of the standard applicable, number of phases
                                            manufacturer"s name, year of manufacturing, type of cooling, connection symbol
                                            rated power, rated frequency, rated voltages, rated currents, short circuit impedance
                                            mass of insulating oil, total mass
online_status:                          this variable indicates if this component is online (true) or offline (false)
substation_ID:                          the identity of the substation this component belongs to */

/*==============================
   local variables declaration
==============================*/

/*==============================
   (local) variables initialization
==============================*/
online_status:= true;                   /* in normal circumstances (at initialization) this component is in operation */

/*==============================
   POOSL behaviour description
==============================*/
sel                                     /* the following methods cannot be executed at the same time. both methods are waiting for
                                        a message. */
                                        /* the first method in which a message is received will be executed */
   handle_offline_requests()()          /* handles offline requests from TMA */
or
   handle_online_requests()()           /* handles online requests from TMA */
les;
init()().                               /* start allover again: infinite behavior */

calc_urgency (analyser_ID: STRING)(urgency: INTEGER)
/*==============================
   method parameters explanation
==============================
substation_ID:                          the identity of the analyzer that requires offline status
urgency:                                the result of this method */

/*==============================
   local variables declaration
==============================*/

/*==============================
   local variables initialization
```

```
==============================*/

/*==============================
    POOSL behaviour description
==============================*/
delay 21;                           /* process time */
if   analyser_ID="Frequency Response Analyser"
then
    urgency:= 2                     /* in case of an FRA (maintenance) the urgency is set at 2 */
else
    urgency:= 4                     /* else at 4 (more urgent) e.g. in case of an alarm */
fi.

switch_off ()(offline_problems: STRING)
/*==============================
    method parameters explanation
==============================
offline_problems:                   problems encountered when going offline */

/*==============================
    local variables declaration
==============================*/

/*==============================
    local variables initialization
==============================*/
offline_problems:= "";              /* at initialization no problems yet */

/*==============================
    POOSL behaviour description
==============================*/
                                    /* this method will shut down all involved components or systems */
                                    /* it may be necessary to warn other objects on the levels below */
                                    /* e.g. when the energy flow needs to be adjusted */
                                    /* after completing all actions: */
delay 1;                            /* component switched off in 1 minute */
offline_problems:="none";           /* there are no complications (in this model offline always succeeds...) */
online_status := false;             /* offline succeeded (in this model offline always succeeds...) */
if   online_status not              /* if offline... */
then
    CEMS!offline_confirmation(component_ID, substation_ID);
                                    /* report the actual action */
else
    offline_problems:= "FAILURE_by_switch_X";
                                    /* an example of a possible offline problem */
  /* do_something()() */            /* investigate (and correct) the problem */
fi.

switch_on ()(online_problems: STRING)
/*==============================
    method parameters explanation
==============================
online_problems:                    problems encountered when going online */

/*==============================
    local variables declaration
==============================*/

/*==============================
    local variables initialization
==============================*/
online_problems:= "";               /* at initialization no problems yet */

/*==============================
    POOSL behaviour description
==============================*/
                                    /* this method will start up all involved components or systems */
                                    /* it may be necessary to warn other objects on the levels below */
                                    /* e.g. when the energy flow needs to be re-adjusted */
                                    /* after completing all actions: */
delay 1;                            /* component switched on in 1 minute */
online_problems:="none";            /* there are no complications (in this model online always succeeds...) */
online_status := true;              /* online succeeded (in this model online always succeeds...) */
if   online_status                  /* if back online... */
then
    CEMS!online_confirmation(component_ID, substation_ID);
                                    /* report the actual action */
else
    online_problems:= "FAILURE_by_switch_X";
                                    /* an example of a possible online problem */
  /* do_something()() */            /* investigate (and correct) the problem */
fi.

handle_offline_requests ()()
/*==============================
    method parameters explanation
```

```
================================*/

/*================================
    local variables declaration
================================*/
lanalyser_ID: STRING;              /* the identity of the analyzer that requires offline status */
approx_duration: INTEGER;          /* approximate offline time (estimated by component) */
maximum_duration: INTEGER;         /* CEMS determines the maximum offline time (e.g. in case of overrunning the
                                   approx_duration) */
offline_problems: STRING;          /* indicates the reason for not going offline, acquired via a message from CEMS */
proposed_delay: INTEGER;           /* CEMS suggests a delay before actually going offline (e.g. to schedule into a load dip) */
target_component: STRING;          /* conditional message reception: checks if CEMS"s response is addressed to this
                                   component */
target_substation: STRING;         /* conditional message reception: checks if CEMS"s response is addressed to this substation
                                   */
trash1: STRING;                    /* used for parameters from received messages with that will not be used (only clarifying the
                                   message) */
trash2: STRING;                    /* used for parameters from received messages with that will not be used (only clarifying the
                                   message) */
urgency: INTEGERI                  /* for priority usage */

/*================================
    local variables initialization
================================*/

/*================================
    POOSL behaviour description
================================*/
TMA?go_offline(analyser_ID, approx_duration);
                                   /* waits for offline_request_messages from TMA */
calc_urgency (analyser_ID)(urgency);   /* first calculate the urgency of the offline request, based on the analyzer_ID */
                                   /* this functionality should be in MAT, instead; I chose TEA for this moment */
CEMS!request_offline_permission (component_ID, substation_ID, approx_duration, urgency);
                                   /* permission must be obtained from the CEMS */
sel                                /* either permission or NO_permission can be the CEMS"s response */
    CEMS?offline_permission (target_component, target_substation, trash1, proposed_delay, trash2, maximum_duration
                        |((target_component=component_ID)&(target_substation=substation_ID)));
                                   /* conditional reception of the message: check addressing */
                                   /* the indices that are good for the readability of the message are moved to trash 1 and 2 */
    delay proposed_delay;          /* follow the advise to wait for some time before actually going offline */
    switch_off ()(offline_problems);   /* switch off and report possible problems */
or
    CEMS?NO_offline_permission (target_component, target_substation
                        |((target_component=component_ID)&(target_substation=substation_ID)));
                                   /* conditional reception of the message: check addressing */
    online_status:= true;          /* component stays online */
    offline_problems:= "no_offline_permission_by_CEMS";
                                   /* problems are identified, to be sent back to TMA */
  /* do_something_else()() */      /* no permission: decide when to do another proposition */
les;
TMA!offline_status(online_status not, offline_problems).
                                   /* returns a status message to the request"s originator */



cluster class transformer(component_ID: String; substation_ID: String)

communication channels
analyser_error, analyser, CEMS

message interface
analyser_error ? error_procedure(STRING);
analyser ? NOT_available(STRING, STRING);
analyser_error ! something_wrong(UNKNOWN, UNKNOWN, UNKNOWN);
CEMS ? NO_online_permission(STRING, STRING, STRING, INTEGER);
CEMS ? offline_permission(STRING, STRING, STRING, INTEGER, STRING, INTEGER);
analyser ? go_back_online(STRING, UNKNOWN);
analyser ! conditions_OK(UNKNOWN);
analyser ! online_again(UNKNOWN);
analyser_error ! abort_now(UNKNOWN, UNKNOWN);
CEMS ! online_confirmation(UNKNOWN, UNKNOWN);
analyser ! conditions_failed(UNKNOWN, UNKNOWN);
CEMS ? online_permission(STRING, STRING, STRING, INTEGER);
CEMS ! request_offline_permission(UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN);
analyser ! analyser_request(UNKNOWN);
CEMS ? NO_offline_permission(STRING, STRING);
CEMS ! offline_confirmation(UNKNOWN, UNKNOWN);
analyser ? available(STRING, STRING, STRING);
CEMS ! request_online_permission(UNKNOWN, UNKNOWN)

behaviour specification
(Oil_Temp_Image: T_oil[temperature/TMA] || Transformer_Energy_Agent: TEA(component_ID,
substation_ID)[energy_maintenance_interface/TMA, energy_management/CEMS] || Transformer_Maintenance_Agent:
TMA(component_ID)[error_channel/analyser_error, analyser_channel/analyser, energy_maintenance_interface/TEA,
```

```
temperature/TEMP])\{energy_maintenance_interface, temperature}[CEMS/energy_management, analyser_error/error_channel,
analyser/analyser_channel]


process class CEMS()
/* no superclass */

instance variables
scheduled_offline_event: Boolean; offline_substation: String; scheduled_offline_component: String; all_equipment_online: Boolean;
scheduled_offline_substation: String; offline_component: String

communication channels
substations

message interface
substations ! NO_offline_permission(UNKNOWN, UNKNOWN);
substations ! online_permission(UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN);
substations ! offline_permission(UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN, UNKNOWN);
substations ? online_confirmation(STRING, STRING);
substations ? offline_confirmation(STRING, STRING);
substations ? request_online_permission(STRING, STRING);
substations ? request_offline_permission(STRING, STRING, INTEGER, INTEGER)

initial method call
init()()

instance methods
init ()()
/*==============================
    instance variables explanation
==============================
all_equipment_online:           boolean is true, when no equipment is offline (used for determining offline permission)
offline_component:              identifier of component that is currently offline
offline_substation:             identifier of substation, containing the offline component
scheduled_offline_component:    identifier of component that is scheduled to be offline (used for higher priority jobs)
scheduled_offline_event:        indicates if there are higher priority jobs that need to be carried out immediately after current
                                job
                                without waiting for a new request
scheduled_offline_substation:   identifier of substation, containing the component, that is scheduled to be offline */

/*==============================
    local variables declaration
==============================*/

/*==============================
    (local) variables initialization
==============================*/
all_equipment_online:= true;        /* in a normal situation, all equipment is online */
scheduled_offline_event:= false;    /* in a normal situation, no offline events are scheduled */

/*==============================
    POOSL behaviour description
==============================*/
par                                 /* parallel execution of two methods */
    watch_offline_requests()()      /* waits for (and handles) offline requests */
and
    watch_online_requests()()       /* waits for (and handles) online requests */
rap.

propose_delay (component_ID: STRING; substation_ID: STRING)(proposed_delay: INTEGER)
/*==============================
    method parameters explanation
==============================
component_ID:                   identity of the component sending the request
proposed_delay:                 delay before actually going offline (e.g. at a load dip)
substation_ID:                  identity of the substation containing the component that sent the request */

/*==============================
    local variables declaration
==============================*/

/*==============================
    local variables initialization
==============================*/
proposed_delay:= 744;               /* later: something between 0 and 1440 (1 day) */

/*==============================
    POOSL behaviour description
==============================*/
delay 2.                            /* just a delay of process time; real behavior to be implemented later on */

check_permission (component_ID: STRING; substation_ID:STRING; approx_duration: INTEGER; urgency: INTEGER)
                 (permission: BOOLEAN; proposed_delay: INTEGER; maximum_duration: INTEGER)
/*==============================
    method parameters explanation
```

```
================================
approx_duration:                    approximate offline time (estimated by component)
component_ID:                       identity of the component sending the request
maximum_duration:                   maximum offline time (e.g. in case of overrunning the approx_duration)
permission:                         true if the requesting component gets permission
proposed_delay:                     suggested delay before actually going offline (e.g. to schedule into a load dip)
substation_ID:                      identity of the substation containing the component that sent the request
urgency:                            for priority usage */

/*==============================
    local variables declaration
==============================*/

/*==============================
    local variables initialization
==============================*/
maximum_duration:= 600;             /* maximum offline time is set to 10 hours */
                                    /* later on: decision depending on the condition of the network, or other parameters */
permission:= false;                 /* not autmatically permission */

/*==============================
    POOSL behaviour description
==============================*/
if  all_equipment_online & (scheduled_offline_event not)
                                    /* permission if component is the first that asks permission */
                                    /* else: the permission stays "false" */
then
    delay 2;                        /* some processing time */
    permission:=true;               /* set permission to "true" */
    propose_delay(component_ID, substation_ID)(proposed_delay)
                                    /* calculate an appropriate delay to propose */
fi.

watch_offline_requests ()()
/*==============================
    method parameters explanation
==============================*/

/*==============================
    local variables declaration
==============================*/
lapprox_duration: INTEGER;          /* approximate offline time (estimated by component) */
component_ID: STRING;               /* identity of the component sending the offline request */
maximum_duration: INTEGER;          /* CEMS determines the maximum offline time (e.g. in case of overrunning the
approx_duration) */
permission: BOOLEAN;                /* true if the requesting component gets permission */
proposed_delay: INTEGER;            /* CEMS suggests a delay before actually going offline (e.g. to schedule into a load dip) */
substation_ID: STRING;              /* identity of the substation containing the component that sent the offline request */
urgency: INTEGERI                   /* for priority usage */

/*==============================
    local variables initialization
==============================*/

/*==============================
    POOSL behaviour description
==============================*/
substations?request_offline_permission (component_ID, substation_ID, approx_duration, urgency);
                                    /* expecting an offline request on the shared channel "substations" */
check_permission(component_ID, substation_ID, approx_duration, urgency)(permission, proposed_delay, maximum_duration);
                                    /* determines permission, proposed_delay and maximum_duration on the basis of request
                                    paramaters */
if  permission
then
    substations!offline_permission (component_ID, substation_ID, "proposed_delay", proposed_delay, "maximum duration",
maximum_duration);
                                    /* message sent back to requester */
                                    /* if the permission message has been sent successfully, then the instance variables are
                                    changed... */
    scheduled_offline_component:=component_ID;
    scheduled_offline_substation:=substation_ID;
    scheduled_offline_event:=true;
                                    /* ...followed by execution of two parallel methods */
    par
        wam_other_substations("going offline within", proposed_delay)();
                                    /* other substations may need to be informed about the coming switching off */
    and
        sel                         /* wait for the actual offline confirmation */
            substations?offline_confirmation (component_ID, substation_ID
                |(component_ID=scheduled_offline_component)&(substation_ID=scheduled_offline_substation));
                                    /* only receive this message if the sender is the scheduled target */
            all_equipment_online:= false; /* not all equipment is still online */
            offline_component:=scheduled_offline_component;
                                    /* no longer scheduled, but actually confirmed */
            offline_substation:=scheduled_offline_substation;
            scheduled_offline_event:=false;
```

```
                                        /* reset variables */
            scheduled_offline_component:="";
            scheduled_offline_substation:=""
        or
            delay (proposed_delay * 1.5) asInteger;
                                        /* if permission message not accepted within this period of time, then timeout and start
                                        ErrorHandling method */
            /* ErrorHandling */
        les;
    rap;
else
    substations!NO_offline_permission (component_ID, substation_ID);
                                        /* if no permission -> confirm that to the component concerned */
fi;
watch_offline_requests()().             /* wait for new requests -> infinite behavior */

watch_online_requests ()()
/*==============================
    method parameters explanation
============================== */

/*==============================
    local variables declaration
==============================*/
lcomponent_ID: STRING;                  /* identity of the component sending the online request */
permission: BOOLEAN;                    /* true if the requesting component gets permission */
proposed_delay: INTEGER;                /* CEMS suggests a delay before actually going online (e.g. to schedule into a load dip) */
substation_ID: STRINGl                  /* identity of the substation containing the component that sent the online request */

/*==============================
    local variables initialization
==============================*/

/*==============================
    POOSL behaviour description
==============================*/
substations?request_online_permission (component_ID, substation_ID
                | (component_ID=offline_component)&(substation_ID=offline_substation));
                                        /* expecting an online request on the shared channel "substations" of the component that is
                                        currently offline */
permission:=true;                       /* always true in this model */
proposed_delay:=14;                     /* always 14 minutes delay proposed */
substations!online_permission (component_ID, substation_ID, "proposed_delay", proposed_delay);
                                        /* report this permission to the component concerned */
par
    warn_other_substations("going back online within", proposed_delay)();
                                        /* other substations may need to be informed about the coming switching on */
and
    substations?online_confirmation (component_ID,
substation_IDi(component_ID=offline_component)&(substation_ID=offline_substation));
                                        /* wait for the actual online confirmation; only receive this message from the offline
                                        component */
    all_equipment_online:= true;        /* reset variables */
    offline_component:="";
    offline_substation:="";
rap;
watch_online_requests()().              /* wait for new requests -> infinite behavior */

warn_other_substations (trash: STRING; proposed_delay: INTEGER)()
/*==============================
    method parameters explanation
==============================
proposed_delay:                         delay before actually going offline (e.g. at a load dip)
trash:                                  used for parameters from received messages with that will not be used (only clarifying the
                                        message) */

/*==============================
    local variables declaration
==============================*/

/*==============================
    local variables initialization
==============================*/

/*==============================
    POOSL behaviour description
==============================*/
delay 2.                                /* just a delay of process time; real behavior to be implemented later on */



process class T_oil()
/* no superclass */

instance variables
```

```
alpha: Real; T_oil: Real

communication channels
TMA

message interface
TMA ! T_oil(UNKNOWN, UNKNOWN);
TMA ? check_oil_temperature(STRING, INTEGER)

initial method call
init()()

instance methods
init ()()
/*==============================
    instance variables explanation
==============================
alpha:                          used in this model to simulate temperature drop
T_oil:                          the current value of the oil temperature */

/*==============================
    local variables declaration
==============================*/
|random_var: RANDOMGENERATOR|   /* used for calculation of alpha */

/*==============================
    (local) variables initialization
==============================*/
random_var:= new(RandomGenerator); /* gives temperature drop a random character */
alpha:= 0.7 + 0.3 * (random_var randomizeSeed random);
                                /* calculation of alpha: varies from 0.7 to 1. probability of temperature drop from 437K to
                                308K */
                                /* (for FRA conditions) within 11 "cycles" will be about 90% (alpha = 0.97). the other 10%
                                times out. */
                                /* later on, the real relation to other quantities (like outside_temperature) can be modeled */
T_oil:=437;                     /* oil temperature, when transformer in operation */

/*==============================
    POOSL behaviour description
==============================*/
watch_temperature_requests()().  /* waits for and handels temperature requests from other object instances */

watch_temperature_requests()()
/*==============================
    method parameters explanation
==============================*/

/*==============================
    local variables declaration
==============================*/
|trash: STRING;                  /* used for parameters from received messages with that will not be used (only clarifying the
                                message) */
max_delay_before_report: INTEGER; /* lays down the maximum time to be waited before responding to the request"s originator */
                                /* and will be used in the case that the temperature drops too slowly (>11 steps of alpha) */
max_T_oil: INTEGER;             /* will be received from the request"s originator: "let me know when T_oil is below this value"
                                */
random_var: RANDOMGENERATOR;    /* used for calculation of alpha */
temperature_problems: STRING;   /* e.g. "temperature timeout" */
temperature_status: BOOLEAN|    /* true if T_oil meets the requirement (below max_T_oil) */

/*==============================
    local variables initialization
==============================*/
random_var:= new(RandomGenerator); /* gives temperature drop a random character */
max_delay_before_report:= 154;  /* 11x the delay of each measurement (14) */
temperature_problems:= "";      /* not yet any problems */
temperature_status:= false;     /* requirements not met, directly after a request (T_oil still 437K by the time of incoming
                                message) */

/*==============================
    POOSL behaviour description
==============================*/
TMA?check_oil_temperature(trash, max_T_oil);
                                /* request from TMA or from similar maintenance agent in other equipment (to be
                                implemented later on) */
                                /* the index "max T_oil", which increases the message"s readable,  moves to trash */
    (while  T_oil>max_T_oil     /* is false, when the requirements are not met */
    do
        T_measurements()();     /* updates T_oil information (adapts T_oil by factor "alpha" in this model) */
    od)
abort                           /* while-loop may be aborted by a time-delay, meaning timeout */
    (delay max_delay_before_report; /* maximum time in which temperature requirement must be met (and a response message
                                should be sent) */
    temperature_status:= false;
    temperature_problems:= "temperature timeout");
```

```
TMA!T_oil(temperature_status, temperature_problems);
                                        /* sends response when (1) condition is met or (2) after a timeout */
T_oil:=437;                             /* resets initial oil temperature to the earlier mentioned "in operation temperature" */
                                        /* later on: the real warming up process of the transformer (or other component) can be
                                        modeled */
alpha:= 0.7 + 0.3 * (random_var randomizeSeed random);
                                        /* calculation of alpha: varies from 0.7 to 1. probability of temperature drop from 437K to
                                        308K */
                                        /* (for FRA conditions) within 11 "cycles" will be about 90% (alpha = 0.97). the other 10%
                                        times out. */
                                        /* later on, the real relation to other quantities (like outside_temperature) can be modeled */

watch_temperature_requests()().         /* wait for new requests -> infinite behavior */

T_measurements()()
/*==============================
    method parameters explanation
================================ */

/*==============================
    local variables declaration
================================*/

/*==============================
    local variables initialization
================================*/

/*==============================
    POOSL behaviour description
================================*/
delay 14;                               /* temperature sample taken every 14 minutes */
T_oil:=T_oil*(alpha).                   /* updates T_oil information (adapts T_oil by factor "alpha" in this model) */


cluster class aClass()

communication channels


message interface


behaviour specification
(EHV_substation_2: substation("Dodewaard2")[energy_management/CEMS] || CEMS: CEMS[energy_management/substations] ||
EHV_substation_1: substation("Doetinchem1")[energy_management/CEMS])\{energy_management}

'
```

Indices

Figures

Tables