Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science
Degree Program of Industrial Engineering and Management

Golan Lampi

Self-Organizing Maps in Decision Support: a Decision Support System Prototype

Master's thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Technology

Espoo, 20th March 2009

Supervisor: Professor Olli Simula
Instructor: D.Sc. (Tech.) Miki Sirola, M.Sc. (Tech.) Jukka Parviainen

HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Program of Industrial Management and Engineering

ABSTRACT OF MASTER'S THESIS

Author: Golan Lampi

Date: 20th March 2009

Pages: viii + 56

Title of thesis: Self-Organizing Maps in Decision Support: a Decision Support System Prototype

Professorship: Computer and Information Science

Code: T-61

Supervisor: Professor Olli Simula

Instructor: D.Sc. (Tech.) Miki Sirola, M.Sc. (Tech.) Jukka Parviainen

Industry and business are full of complicated decision making processes in which there is a high probability of human error. These processes are most crucial in the operation of nuclear power plants. The quality of decisions can be increased and probability of errors can be reduced by providing computerized decision support for the decision maker.

Self-Organizing Map (SOM) is a useful method for visualizing high-dimensional and large datasets. The aim of this work is to find approaches for using SOM in supporting the decision making processes of nuclear power plant operators, and to analyze whether these approaches can be used for decision support in other applications. The research methods are prototyping, data mining and survey of literature.

A prototype of a decision support system (DSS) platform (DERSI) has been developed. The prototype uses a collection of methods for decision support, including SOM quantization error, SOM U-matrix, fuzzy logic, rule-based reasoning and case-based reasoning. It is programmed with the Matlab programming language and uses a SOM Toolbox add-on. It has a graphical user interface that contains visualizations of the methods.

Two units of a DSS prototype have been built on this platform. One uses data from a Simulink simulation model and the other unit uses data from the Teollisuuden Voima (TVO) nuclear power plant simulator. These prototype units demonstrate the potential of the prototype methods. Other approaches for using SOM in decision support were found from literature. The thesis compares these approaches with the prototype methods and discusses the possible use of this prototype in other applications.

Keywords: Data Mining, Decision Support, Self-Organizing Map (SOM), Rule-Based Reasoning, Fuzzy Logic

TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta
Tuotantotalouden tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Golan Lampi

Päiväys: 20. maaliskuuta 2009

Sivumäärä: viii + 56

Työn nimi: Itseorganisoituvat kartat päätöksenteon tuessa: päätöksenteon tukijärjestelmän prototyyppi

Professuuri: Informaatiotekniikka

Koodi: T-61

Työn valvoja: Professori Olli Simula

Työn ohjaaja: TkT Miki Sirola, DI Jukka Parviainen

Teollisuus ja liiketoiminta ovat täynnä monimutkaisia päätöksentekoprosesseja, joissa inhimillisen virheen mahdollisuus on suuri. Nämä prosessit ovat hyvin kriittisiä ydinvoimaloiden toiminnan ohjauksessa. Päätöksien laatua voidaan parantaa ja virheiden todennäköisyyttä vähentää antamalla tietokoneistettua päätöksenteon tukea päätöksen tekijälle.

Itseorganisoituva kartta (SOM) on hyödyllinen tapa visualisoida moniulotteisia ja suuria data-aineistoja. Tämän työn tavoitteena on löytää tapoja SOM-menetelmän hyödyntämiseen ydinvoimalan operaattorin päätöksenteon tuessa ja analysoida, voiko kyseisiä tapoja käyttää myös muiden sovellusalueiden päätöksenteon tukeen. Työn tutkimusmenetelmät ovat kokeellinen prototyyppikehitys, tiedonlouhinta ja kirjallisuustutkimus.

Tutkimuksessa on toteutettu prototyyppi (DERSI) päätöksenteon tukijärjestelmän (DSS) alustasta. Se hyödyntää päätöksenteon tuessa kokoelmaa erilaisia menetelmiä, kuten SOM kvantisointivirhettä, SOM U-matriisia, sumeaa logiikkaa, sääntöpohjaista päättelyä ja tapauspohjaista päättelyä. Prototyyppi on ohjelmoitu Matlab-ohjelmointikielellä ja se hyödyntää Matlabin SOM Toolbox -laajennusta. Siihen kuuluu myös graafinen käyttöliittymä, joka sisältää käytettyjen menetelmien visualisoinnit.

Tutkimuksen alustalle on rakennettu kaksi päätöksenteon tukijärjestelmän prototyyppiyksikköä. Yksi niistä hyödyntää tutkimuksen Simulink-simulaatiomallin dataa ja toinen Teollisuuden Voiman (TVO) ydinvoimalasimulaattorista saatua dataa. Nämä yksiköt demonstroivat prototyypin menetelmien mahdollisuuksia. Kirjallisuudessa esiintyi myös vaihtoehtoisia tapoja hyödyntää SOM-menetelmää päätöksenteon tuessa. Näitä verrattiin prototyypin menetelmiin ja lisäksi pohdittiin, voiko prototyyppialustaa hyödyntää muilla sovellusalueilla.

Avainsanat: tiedonlouhinta, päätöksenteon tuki, itseorganisoituva kartta (SOM), sääntöpohjainen päättely, sumea logiikka

Foreword

This Master's thesis has been done in the Department of Information and Computer Science in Helsinki University of Technology in 2008-2009. Although the background work of the thesis begun already in the summer 2003 when I came to work for the first time to the department.

The industrial partner in this project was Teollisuuden Voima Oy (TVO) in Olkiluoto, Finland. The project has been funded by TEKES and TVO. I am grateful to both for supporting and financing the project. The project has been part of IDE group in the beginning and part of NOTES project in the end.

I am very grateful for my supervisor professor Olli Simula and both instructors Miki Sirola and Jukka Parviainen for guidance, support and most of all patience in a process that has taken much longer than was intended. A lot of unexpected events and obstacles in my personal life have desperately tried to prevent me from writing this thesis but fortunately I have been able to finally complete it. I am forever grateful for Miki from persistent support and belief in me in times when even I did not believe in myself. Few instructors would have so much patience.

Thanks for my colleagues Jaakko, Mikko and Janne, Risto and Teemu from support and from being excellent room mates. Thanks for Entropy and Rose Garden, and my very unreliable computers for providing content to my life when I was not working :)

Most of all, I would like to thank my family. My parents Juha and Avigail, sisters Johanna and Pamela, our dog Mona and my love Maikku. And two dear friends Mipi and Roni that have been friends since childhood.

Otaniemi, Finland, March 20th, 2009

Golan Lampi

Contents

List of Figures

Acronyms and Terms

AHP	Analytic Hierarchy Process
BI	Business Intelligence. DSS is often referred to as BI in business context.
BMU	Best Matching Unit. BMU of a vector in a neural network is the neuron that has the smallest distance to the vector.
BWR	Boiling Water Reactor
CRISP-DM	CRoss Industry Standard Process model for Data Mining. Data mining activities have been standardized into this standard, whose latest version is 2.0.
DECS	DECS is a Matlab struct that contains all data of a DERSI unit.
DERSI	DERSI platform is abbreviated as DERSI.
DERSI Platform	DERSI platform refers to the DSS prototype platform of this study. The platform is application independent. The name of the platform comes from DEcision support Recommendation System.
DERSI Unit	DERSI unit refers to a particular instance of DERSI Platform that is customized and build for a particular application and dataset. There are two DERSI units in this study: Simulink unit and TVO unit.
DSS	Decision Support System. DSS is called in some contexts with other names like BI, Management Information System (MIS) or Executive Support System (ESS).
ESS	Executive Support System
GUI	Graphical User Interface
MIS	Management Information System
MMI	Man-Machine Interface. MMI refers to the part of a DSS that is connected to a human user.
OLAP	OnLine Analytical Processing. OLAP is a technique to quickly answer multi-dimensional analytical queries. Various quantities like sums, averages and correlations are deduced from a multi-dimensional dataset.
PCA	Principal Component Analysis. PCA is a method of statistics.
SOM	Self-Organizing Map. SOM is an unsupervised neural network algorithm.
SOP	Standard Operating Procedures
SQL	Structured Query Language

TVO	Teollisuuden Voima Oy. TVO is a nuclear power plant company that is an industrial partner of this thesis.
UI	User Interface
WWW	World Wide Web

Chapter 1

Introduction

Fast development of technology, fierce competition in markets and demand for safer systems than ever are reasons why enormous amount of research effort are put on data analysis and visualization methods. Fast growth of processor power, memory, disk storage and network capacity create new opportunities for information systems.

This study uses two groups of previous studies as a basis. First is a doctoral thesis that has been conducted of utilizing rule-based reasoning in failure management of safety critical processes [1]. Second is a series of studies where self-organizing map (SOM) is utilized for industrial process monitoring [2, 3, 4, 5, 6, 7, 8]. As SOM is one of the most used neural network algorithms and has been invented in author's research institute, a lot of SOM research is conducted there. When researchers in these two fields collaborated the following hypothesis arouse: combination of rule-based reasoning and neural methods can be utilized in process monitoring decision support.

It was decided to build a decision support system (DSS) [9] prototype that tries to prove the hypothesis and demonstrate the strengths of these methodologies and their combination. Many of industrial processes are safety critical. Nuclear power plant process monitoring was chosen as the application for the prototype because strong contacts and experience of the field existed in the research group steering this study. A challenge in developing tools for this application is that the application is extremely safety critical and new technology is tested and utilized very conservatively in such applications. The reason is that in safety critical applications the replacement of technology with newer one is considered to be an expensive risk. The scope of the study was extended to identify other applications where the prototype could be utilized.

Nuclear power plant process is monitored constantly by an operator or team of operators whose task is to ensure efficient and safe operation of the plant [10]. They use various systems for monitoring the process and preventing faults. DSS is a computer system whose purpose is to make the decision making process of the system user faster, more consistent, accurate, fault free and safe. Prototyping is a methodology for interactive development of a system [11]. It is used for experimenting new methods and structures in situations were feedback is required constantly. Prototyping can have many iterations of development and feedback gathering.

SOM is one of the most interesting methods for two-dimensional visualization of

data [12]. Large amount of publications has been written of SOM and its applications. In many applications computational efficiency is one of the primary concerns and this is one of the most important strengths of SOM [13]. SOM is used in many different ways in this study, and one of them is classification. It can be used for case-based reasoning, where an analogy is drawn from cases similar to the current situation.

Logical statements like A IS B or IF A THEN B are useful way to transfer human knowledge into a computerized form that can be used automatically. Rule-based reasoning is a methodology for the computer to process this knowledge and create new facts from existing ones [14]. As the most of real life knowledge is fuzzy in nature, fuzzy logic can be useful in coding the process knowledge [15, 16].

The largest problem with rule-based reasoning is that in almost any application a very large amount of rules are required for coding all necessary knowledge and it is difficult to be sure that nothing essential is missing from the rule database. By combining rule-based reasoning, case-based reasoning and fuzzy logic it is possible to reduce the amount of required rules to minimum and code the knowledge in the language where it has the most simple representation.

This study is funded by TEKES and Teollisuuden Voima (TVO) nuclear power plant and the study is a part of NOTES project.

1.1 Research Problem, Objectives and Scope

Study of this thesis is very interdisciplinary. Research fields connected to the study are decision support, data mining, neural networks, software engineering, graphical user interface (GUI) development, safety engineering, system engineering, quality management, artificial intelligence, business intelligence and operations management. The research problem is formulated as follows:

- **How can self-organizing map (SOM) be utilized in nuclear power plant operator decision support?**

The following research questions help solving the research problem.

- How is information system designed?
- How are system models, algorithms and data structures selected?
- How are man-machine interface (MMI) visualizations selected?

These research questions are answered only partially in this thesis. They are intended as a support for solving the research problem. While performing the study knowledge of SOM utilization in decision support of other applications was also acquired and documented.

The scope of the study has been limited by focusing only on SOM algorithm and rule-based reasoning in the application of nuclear power plant decision support. Other neural and reasoning methods are left outside the scope. Benchmarking of the prototype has been left outside the scope because it is difficult. But some subjective and intuitive evaluation has been performed.

While performing the study the scope was first extended by exchanging rule-based reasoning into fuzzy rule-based reasoning. After that it was again extended by studying if it is possible to use the prototype of the study in some other application than nuclear power plant operator decision support.

The motivation behind this study is to increase safety and efficiency in process control. As failure in nuclear power plant is very expensive, safety is a critical issue. The study is a part of a larger problem of how can data-based learning methods increase safety and efficiency of DSSs and how can data material be utilized as efficiently as possible in decision support and process control. Important goals in nuclear power plant operator decision support are early fault detection, identification and repairing.

1.2 Structure of the Thesis

The thesis is organized as follows. Chapter 2 explains decision support, decision support systems, safety critical systems and related concepts. Chapter 3 explains software engineering, graphical user interface (GUI) design and data mining briefly. Chapter 4 describes the most important methodologies of the study: self-organizing map (SOM), rule-based reasoning and fuzzy logic. Chapter 5 is the most important chapter. It describes nuclear power plant process and DERSI, the constructed decision support system platform prototype. Results are presented and evaluated in Chapter 6. Chapter 7 has summary and conclusions.

Chapter 2

Decision Support

2.1 Definition of System and Process

System can be defined in many ways. One definition is as follows: a system is a purposeful collection of interrelated components that work together to achieve some objective [11, p. 21]. Systems are separated into three distinct parts: *inputs*, *processes* and *outputs* [9, p. 35]. A diagram of a system in the context of decision support is shown in Figure 2.1.

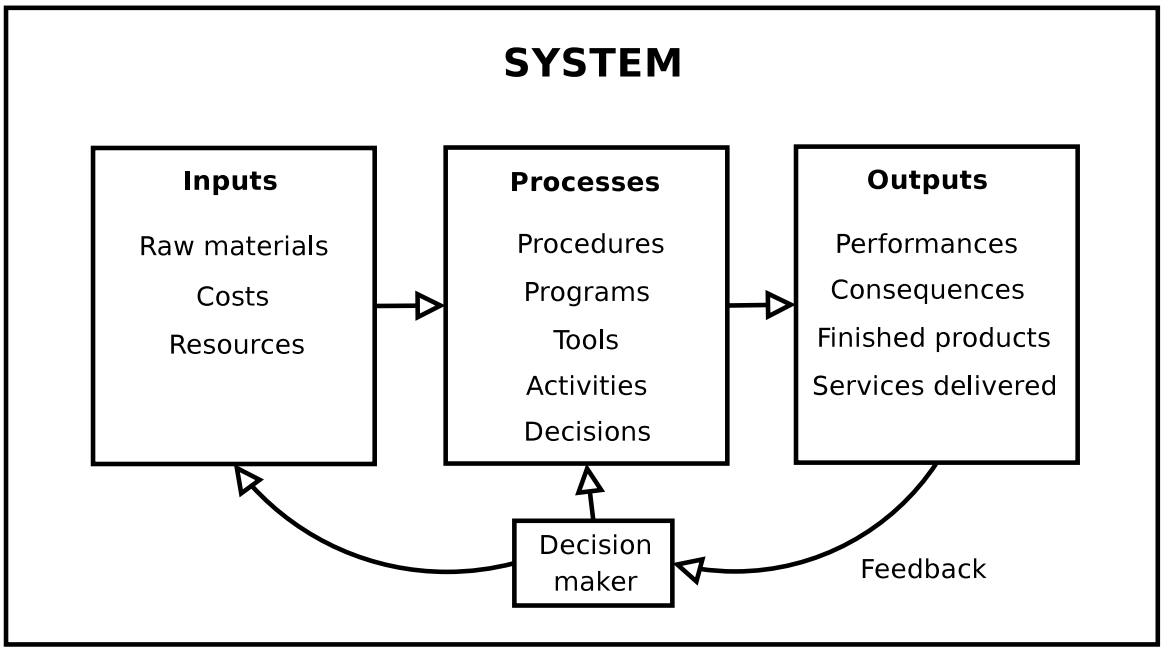

Figure 2.1: Processes, inputs, outputs and the decision maker in a system.

Inputs are the elements that enter into the system. Examples of inputs are patients admitted to a hospital, raw materials entering a chemical plant or data input into a computer or process control system. *Processes* are all the elements that are needed to convert the inputs into outputs. For example a process in a chemical plant may include elements like heating the materials, using operating procedures and using a material

handling subsystem. In a hospital a process may include elements like conducting tests and performing surgeries. *Outputs* can be for example finished products or the consequences of the running process. Gasoline is one output of a chemical plant and cured people are an output of a hospital.

2.2 Decision Making Processes

One important process type is the *decision making process*. A lot of decisions need to be made, either by people or automation. If automation is making decisions, people have decided decision logic of the automation in advance. Decision making process in problem solving can be divided into four phases: *intelligence*, *design*, *choice* and *implementation*. Actions in the phases are summarized in the following list [9]:

- Intelligence
  - Finding the Problem
  - Problem Classification
  - Problem Decomposition
  - Problem Ownership
- Design
  - Modeling
  - Principle of Choice
  - Developing Alternatives
  - Predicting Outcome of Alternatives
- Choice
  - Search of Alternatives
  - Evaluation of Alternatives
- Implementation

The first part of intelligence phase is *finding the problem*. It is important to distinguish between a symptom and a problem. For example a symptom could be high temperature in a nuclear reactor and a problem could be a broken valve. The phase of intelligence is a primary target for decision support systems (DSS) designed for non-structured problems [9, p. 60]. *Models* of design phase describe how symptom variables change in relation to control variables and how uncontrollable variables act. There are many approaches to model the situation of the decision making. Some of them are game theory, scenario analysis, differential equations and various optimization methods. *Principle of choice* is the principle of how risk averse or risk seeking the decision should be. *Alternatives* are possible courses of action.

If the amount of alternatives is large, the search for effective alternatives in choice phase can be complex. A few important approaches for evaluating alternatives are

utility theory, analytic hierarchy process (AHP), sensitivity analysis and what-if analysis. Implementation means carrying out the decision. In business context this could mean communication, explanation and justification of the decision. In nuclear power plant decision support context this could mean opening or closing of various valves of the plant.

Structured processes are routine and repetitive problems that have standard solutions. Unstructured processes are fuzzy and complex problems that have no simple solutions. Decision making processes can be classified as structured, semi-structured or unstructured. Semi-structured means that some parts of the process are structured and some are not. One typical feature of an unstructured decision situation that the amount of alternative actions is very large. Structured and some semi-structured decisions have been supported by computers for many decades, especially decisions of the operational and managerial control type. Such decisions are made in all functions of an organization, especially in finance and operations [9, p. 13].

Problem finding is in very important role in many decision making processes. Some of those processes do searching with problem knowledge that is in a hierarchical form. Such processes are used in many diagnostic applications, for example when doctors performs diagnoses for patients and computer support personnel perform diagnoses while repairing a computer system [17]. The problem solver begins from the root of a knowledge hierarchy, asks a question, finds an answer for it and depending on the answer chooses a node of the hierarchy. Then it repeats the procedure until a leaf is chosen.

2.3 Quality Management Process

Quality management is a process that is present in all functions of an organization. The goal of the process is the same in every organization so it has been possible to standardize the process. *ISO 9000* is a standard that defines quality management procedures. *ISO 9000* certified organizations conform to this standard. Certification is a sign of quality for those evaluating the organization. Evaluators could be investors, users of a product or other stakeholders. Safety is an important part of quality in a nuclear power plant process. [18]

Quality management can be divided into *quality control*, quality assurance and quality improvement. It can also be divided into design quality, production quality and customer quality. Design quality is the relation between the intentions of the product designer and the actual product design. Production quality is the relation between the production process specifications and the actual process output. Customer quality is the relation between the customer needs and the specifications of a product.

The purpose of quality control is to ensure that the outcome of production process is what was intended. Each of different production processes have their own quality control activities. For example cell phone production quality is controlled both by measurements in production process and feedback information. The first can be statistical samples of defects in a cell phone batch and the second can be the amount of warranty returned phones or the amount of negative customer feedback.

A part of the quality control process is mental models for finding reasons of faults

in the production process. Such mental models are for example brainstorming, diagnosing methodologies, root cause analysis, standard operating procedures (SOP) and fishbone analysis. One of the most important tools of quality control is the control chart that is a time series plot of a production process variable [19, p. 1128]. It is associated with upper and lower warning and alert limits. If the variable is a statistical quantity, monitoring of the charts is calle statistical quality control.

2.4 Systems Engineering

The science of constructing systems is called systems engineering and it is very interdisciplinary field. The field is called with many other names like industrial engineering, production engineering, operations management, manufacturing systems engineering and control engineering. Systems engineering is connected to many other engineering disciplines like software engineering, enterprise systems integration, intelligent system engineering, expert system engineering, artificial intelligence engineering and data mining. [20, 9]

The most important systems engineering projects are process re-engineering projects. Processes of an existing organization are mapped with business process mapping. Various process maps and process flow charts are created. These maps have details of both the information flow and the material flow of the organization. The maps and charts are analyzed with methodologies like flow analysis and new improved process maps are acquired. After this both the form of the organization and its technological infrastructure are re-engineered. [21]

2.5 Decision Support Systems

Various systems are build for running, automating and supporting the processes of an organization. These systems are networks of people, infrastructure and knowledge that are connected together. Important infrastructure are computers and manufacturing machinery. Knowledge can be either explicit or tacit. It can be either in memory of people like the knowledge of a handcraft shoemaker. Or it can be saved in form of written or electronic documents like emergency procedures and rules of a nuclear power plant. The knowledge can be either in structured or unstructured form. [22]

Many human-computer interfaces emerge in these systems. They are called *man-machine interfaces* (MMI). In business decision support context they are called dashboards. MMIs require a lot of attention as a bad MMI is a major reason of human errors causing defects, faults and non-optimal operation of a process. [1, 9]

If the system provides support for a human user, it called a *decision support system* (DSS). In some contexts the word is reserved for a system devoted for unstructured problem solving. DSSs of some applications have different names like management information system (MIS) or executive support system (ESS). Business decision support is often called *business intelligence* (BI). Most decision support systems use a data warehouse as an input. Data warehouse is a database that collects data from all other databases of an organization. Data mining is an essential part of modern decision support and is often applied directly to the data warehouse. Mathematical queries,

reports, statistical analysis and online analytical processing (OLAP) are possible with the data warehouse. OLAP is simple data mining that calculates summarized quantities like averages and sums from multi-dimensional data cubes.

Many names are used to refer to the same computer or manufacturing systems. The used name depends on which engineering discipline is used to model and analyze the system. Some of these names are *information system*, *enterprise system*, *control system*, *decision support system* and *expert system*. System structures in the organization depend on the form of the organization. Different structures would emerge in a bureaucratic organization than in a flexible and organic organization [23].

2.6 Safety Critical Systems and Cost of Quality

Cost of quality can be divided into four groups: 1. *prevention* of failures, 2. *appraisal*, 3. *internal failure* and 4. *external failure*. Appraisal cost is created from inspecting or monitoring. Internal failure cost is created from sales loss of defective products that need to be discarded. External failure cost is created from defective products that have been already sold to a customer. It can be from warranty service or from reclaims for compensation. In some processes there are only failure costs that cannot be separated to internal and external parts. Examples of such processes are production of services and nuclear power generation. Usually it makes sense to minimize the total quality cost that is sum of the four costs. [18]

In some processes the cost of failure is much higher than the cost of prevention or appraisal. These processes are *safety critical*. Examples of such processes are nuclear power production, airplane traveling and diagnosing of fatal diseases. Safety engineering is a field specialized into design of fault-free systems that are required for control and support of safety critical processes. Very often safety requires security so safety and security engineering are closely connected. [24]

Safety critical systems are designed to be redundant and they have to go through much more detailed analysis than traditional computer or production systems. The manufacturing of Olkiluoto 3 power plant will last for almost 10 years because of high safety requirements [25]. Safety critical systems in nuclear power plants are built for *defense in depth*. There are many subsystems in different levels and if one system malfunctions, the next system becomes operational. Every subsystem is independent of each other and the probability of all subsystems failing simultaneously is extremely small [1]. That probability is the product of all individual subsystem failure probabilities. When creating a safety critical computer program its reliability can be increased by using methods of propositional logic to prove that the program is fault free [26].

In safety critical applications a false negative diagnosis of a fault is much more expensive than a false positive diagnosis. So after optimizing the operation of a safety critical system it is likely that the system produces false positive diagnoses easily. [27].

Chapter 3

Software Development and Data Mining

3.1 Software Engineering

There are many models for the process of a software project. The most simple and traditional software process is the *waterfall model* that consists of five sequential phases: 1. requirements definition, 2. system design, 3. implementation, 4. integration and 5. system operation [11]. Opposite of waterfall model is *agile programming* process that is very flexible and organic [28]. It does not have distinct phases.

The most important part of a software project is requirements engineering. If the requirements are poor, the outcome of the programming will be poor. Requirements engineering has four parts: 1. feasibility study, 2. requirements elicitation and analysis, 3. requirements specification and 4. requirements validation. Requirements can be divided into functional, non-functional and domain requirements.

In requirements elicitation the knowledge and opinions of project stakeholders are transferred into a written form. Cross-functional teams are formed from customers, engineers, programmers, data analysts or other experts or stakeholders. Brainstorming, meetings and interviews are important methods in requirements elicitation. Marketing research methods like questionnaire surveys, phone or Internet queries can also be utilized [29]. In some projects useful data can be found by conducting structured query language (SQL) or online analytical processing (OLAP) queries to related databases. *Simulations* can be developed to get more understanding of the software users problem. Examples are differential equation systems formed from laws of physics, probabilistic Monte Carlo simulations or game theoretic simulations [9].

Rapid prototyping is a process of using a prototype for getting feedback and requirements from system users. The process is iterative and consists of many development cycles. In every cycle a new version of the prototype is created and feedback is collected from a prototype user. After requirements of the software project are clear the prototype is discarded and the actual system is developed in different programming environment.

3.2 GUI Design and Visualizations

Graphical User Interface (GUI) or man-machine interface (MMI) design is a field of its own [30]. In a safety critical application like nuclear plant process monitoring it is especially important that the user interface is of good design. The amount of relevant information should be maximized and the amount of irrelevant information should be minimized in every GUI screen. Psychological and cognitive factors of the GUI user have to be taken into account. The interface should be easy to use and have understandable results. [1]

Visualizations of the study can be divided into five groups: text, plots, maps, on/off visualizations and bar visualizations. Text is one of the most common ones and it can be either in structured or unstructured form. *Plot* is a graphical representation that has one variable in x-axis and another one in y-axis and every point in x-axis is associated to only one point in y-axis. If time is on x-axis the data of the plot is called time series. Value on the y-axis is the value of the time series plot. Plots are used for example in industrial process monitoring. Time series plots in quality control are referred to as *control charts* [19, p. 1128]. Control chart plot has warning and alert lines drawn above and below the area where the plot is supposed to be under normal operation. If the value of the plot variable increases or decreases so that the plot goes beyond the lines, the operator is notified.

There are numerous types of *map* visualizations. A map is a two-dimensional area that has elements organized into some formation. One type of map is a diagram model of a nuclear power plant that the operator can use for choosing a plant component for more detailed analysis. Another type of map is a 2D projection of a multi-dimensional data set. Such projections are for example *principal component analysis* (PCA) projection, curvilinear projection and *self-organizing map* (SOM) [31].

An example of an *on/off visualization* is a lamp that signals a process state by being on, off or by blinking. Car and airplane dashboards are full of such visualizations. *Bar visualizations* are similar to plot visualizations, but only one point of the plot is expressed with the bar. Bar height is mapped to a data variable value. Examples of bar visualizations are described in Sections 5.5.8 and 5.5.9.

Expression of time is a significant factor in visualizations. In time series plots one spatial dimension is assigned for expressing time. In a SOM a vectorial time series can be visualized with a trajectory where the timeline becomes a two-dimensional path. In animation time itself is assigned for expressing time.

3.3 Data Mining Process

Data mining process has been standardized. The name of this standard is *CRoss Industry Standard Process model for Data Mining* (CRISP-DM) [32]. It has six phases: 1. *business understanding*, 2. *data understanding*, 3. *data preparation*, 4. *modelling*, 5. *evaluation* and 6. *deployment*. The most important part of *business understanding* is the definition of the problem. The most important part of *data understanding* is exploratory data analysis and statistical analysis. Simple statistics can be used and even more sophisticated methods like principal component analysis (PCA) projections

or unsupervised learning. Some statistical methods are calculation of mean, variance, cross correlations, histograms and spectrum analysis. [31].

Important part of *data preparation* is data preprocessing. Data is filtered, variables are selected or discarded, samples are selected or discarded and features are extracted with transformations. Choice of teaching samples and variables is significant when teaching neural models like SOM [33]. *Curse of dimensionality* means that it is difficult to build accurate models of a very high-dimensional dataset [31]. So the dimension of the data has to be reduced with variable selection or projections. Common projections used for this purpose are PCA and random projection [12, 13]. The most important *modeling* concepts in data mining are description, classification, regression and association. Description refers to statistical models, clustering, and other kind of unsupervised learning. In classification the dataset is divided into classes, and models for identifying samples of different classes are constructed. In regression a model for calculating one variable (*dependent variable*) from other variables (*independent variables*) is constructed. Association is the learning of association rules from the dataset. Association learning finds causal relations between different variables.

In *evaluation* the performance of the models is evaluated with scoring functions. The goal is to find a such model that maximizes the score value. Deployment means using the model of the data mining process in production. Computational efficiency of the chosen model or method is an issue when mining very large datasets [31]. Before starting the data mining appropriate tools need to be chosen. Commonly used tools are Matlab [34], SPSS and Excel computer programs.

Chapter 4

Methodologies

4.1 Self-Organizing Map (SOM)

The Self-Organizing Map (SOM) is an artificial neural networks algorithm that is based on competitive learning [12]. It can be used for many purposes, for example, clustering, vector quantization and 2D visualization of multi-dimensional data. The most important strength of SOM is its computational efficiency. In many applications the SOM algorithm is fast compared to alternatives [13]. It is also very good to recognize noisy or ill-defined data [12, p. 64]. A weakness of SOM is that it forgets time relations of samples when learning [12]. There are ways to overcome this weakness [35] and one of them is to create delayed or difference versions of original teaching variables into a teaching dataset of the SOM.

4.1.1 Algorithm

In SOM neurons are placed at the nodes of a lattice that is usually one or two-dimensional. Higher-dimensional maps are possible but uncommon [33, p. 443]. The neurons are prototype vectors that have the same dimensionality as the input data. They are initialized either randomly or with some other initialization scheme [8].

The SOM teaching algorithm is illustrated in Figure 4.1 and explained as follows. The algorithm is iterative. First the prototype vectors are initialized. After that an iterative teaching phase begins. A SOM is trained with input data. As an input data vector is fed to the SOM the best matching unit (BMU) is searched from the prototype vectors. The BMU, denoted here by b_i, of an input data vector x_i is the map unit with a prototype vector m_j closest to x_i [8, p. 12]:

$$b_i = \arg\min_j \{\|x_i - m_j(t)\|\} \tag{4.1}$$

Here i is the input data vector index, j is the map unit index and t is the training step index. There are many definitions for distance between two vectors but Euclidean distance is a common one [12, p. 16]. If the input data vector x_i has missing variable values, those variables are ignored in the distance calculation and in the subsequent update step. After this the BMU and its neighbor cells are updated by moving their prototypes towards x_i so that new values for the prototype vectors are:

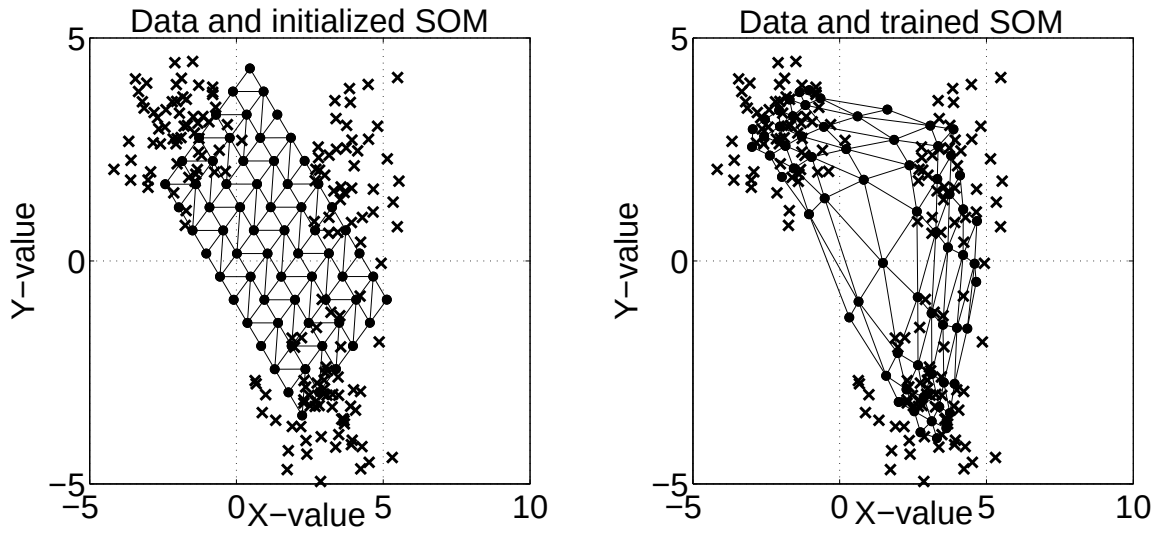

Figure 4.1: Two-dimensional SOM in input space. Crosses are input data vectors and dots are SOM prototype vectors. In the left picture is an untrained map. In the right picture is a trained map. It can be seen how the SOM approximates the input data distribution.

$$m_j(t+1) = m_j(t) + \alpha(t)h_{b_ij}(t)[x_i - m_j(t)] \tag{4.2}$$

$\alpha(t)$ is learning rate and $h_{b_ij}(t)$ is a neighborhood function centered on the winner unit. The neighborhood function gets its maximum value in the winner unit and decreases monotonically with increasing distance on the map grid. The rate of this decreasing depends on neighborhood radius $\sigma(t)$. [8]

During training both the learning rate $\alpha(t)$ and the neighborhood radius $\sigma(t)$ decrease monotonically and the SOM prototype vectors become euclidically close to the input data vectors. The SOM algorithm seeks to teach the map so that the *quantization error* of the input data is minimized while the map still preserves the topology of the input space. The quantization error of the map with input data vectors x_i can be calculated with the following formula:

$$E_q = \sum_{i=1}^{N} \|x_i - m_{b_i}\|^2 \tag{4.3}$$

N is the amount of data samples and m_{b_i} is the prototype vector of the BMU of x_i. After the map is trained, it can be used to identify data that is similar to the teaching data by using that data instead of input data vectors in the quantization error calculation. Data that is different than the teaching data, gets a larger quantization error than data that is similar to the teaching data [36].

4.1.2 SOM Visualization

SOM component plane,U-matrix and trajectory visualizations are shown in Figure 4.2. A SOM component plane is a two-dimensional lattice that contains either rectangular or hexagonal cells. The color of each cell represents a value of a prototype vector variable. Each variable has a separate plane and by comparing the planes with each other it is possible to find correlation relationships between variables.

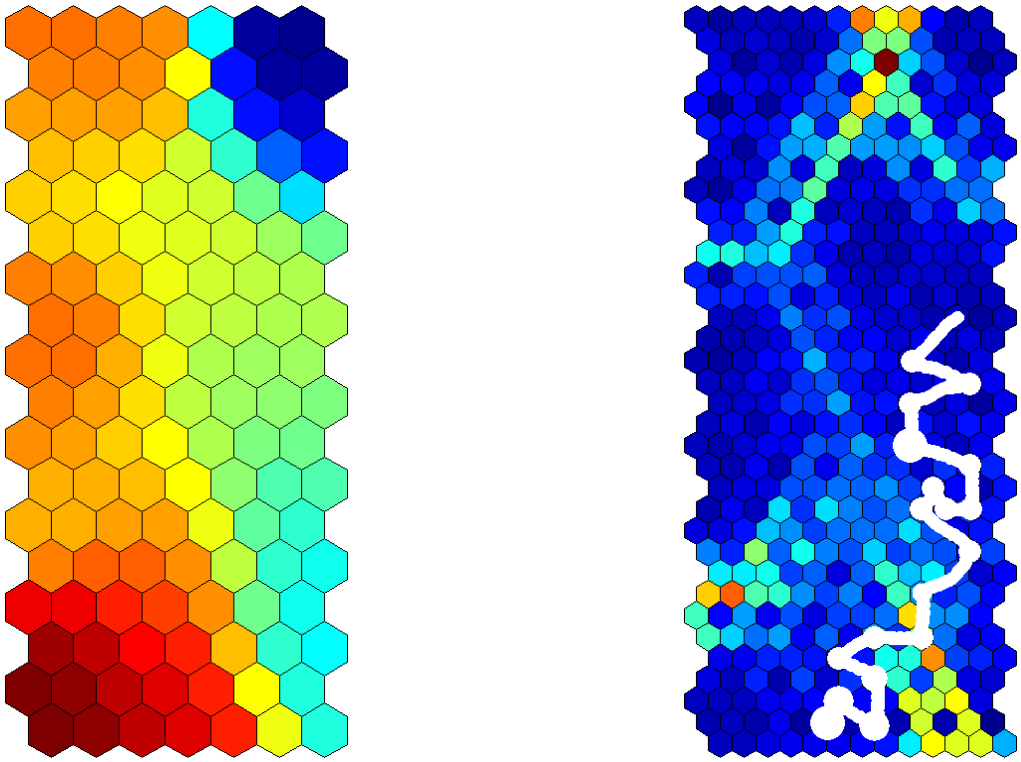

Figure 4.2: SOM component plane visualization is shown on the left and SOM U-matrix visualization on the right. A SOM trajectory is plotted into the U-matrix visualization.

U-matrix is similar to a component plane but there is only one U-matrix for one SOM and in the U-matrix there is one cell per every contact between two closest neighbor prototype vectors of the map. Every prototype vector that is not in the border of the SOM has six cells in the U-matrix when the SOM is of hexagonal type. The color of a cell represents the distance between the two prototype vector neighbors. Clusters can be easily seen in a U-matrix visualization as the cluster borders are shown in brighter colors.

A time series can be visualized by drawing a trajectory into a SOM U-matrix or component plane visualization. A BMU of a vector of a time series is marked with a point. Then a BMU of the next vector of the time series is also marked with a point and a line is drawn from the previous BMU to the current BMU. This procedure is repeated until BMUs of all time series vectors are drawn. A curve showing the time series propagation in the SOM is formed.

4.1.3 SOM as Regression Model

SOM can be used as a nonlinear regression model [37]. If a variable of a teaching data vector is not used in distance calculation when finding its BMU, the SOM becomes a regression model for that variable. If the variable is time, SOM can be used to find out the time value of the teaching data vector that is similar to a test data vector. In this study that value is called *SOM progress of the test data vector on the teaching data*. Finding of the desired time value works only in limited amount of situations. In process monitoring application, if the test data vector represents the current process state and teaching data is a time series portion from fault state X, the value is called *current state SOM progress on fault state X* or just *SOM progress on fault state X*. There has been preceding studies of the potential use of the SOM progress value [38].

4.2 Rule-Based Reasoning

One way for saving knowledge are rules based on propositional logic. Such rules are IF-THEN rules where IF part is called *antecedent* and THEN part is called *consequent*. Both the rule antecedent and the consequent can be A IS B type of logical statement. The consequent can also be DO X type of imperative. Consequents of some rules can be antecedents of other rules. The rules are contained in a hierarchical *rule base* having many rule trees. Root of every tree is an imperative, nodes and leaves are logical statements and branches are rules. Appendix A contains an example of a simple rule base.

The *rule base* is a dedicated database. It is too inefficient to test all of the rules of a large rule base. That is why the rules are tested in an *inference* process. It begins either from leaves of the trees (forward chaining inference) or roots of the trees (backward chaining inference). The testing process propagates to that direction in the tree where the rule antecedents get true values. More details of the rule-based reasoning process can be found from [39].

4.3 Fuzzy Logic and Reasoning

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise [15, 16]. In fuzzy set theory an element can be a partial member of a set and its membership value can range between 0 and 1. The truth value of a fuzzy logic statement can also range between 0 (false) and 1 (true). Such statement has a membership function that calculates the statement truth value from some related quantities. This process is called *fuzzification* and it is the first of three phases in the operation of a fuzzy reasoning system.

The second phase is fuzzy *inference*. All fuzzy rules, their statements and logical operations (AND, OR, NOT) of antecedents are processed. Usually AND of two truth values chooses the higher one and OR chooses the lower one. After processing the logical operations their outputs are assigned for the consequents.

The third phase is *defuzzification*. It is the process of transforming consequent truth values so that each consequent quantity is assigned only one result value. A

consequent quantity is assigned to many fuzzy statements, membership functions and truth values, so the result value calculation can be complex, especially if accurate techniques like centroid are used. In some applications the consequent truth values are the desired output and defuzzification is not required.

Chapter 5

DERSI Platform Prototype

5.1 Nuclear Power Plant

A nuclear power plant produces electricity with a nuclear fission process [40]. Structure of a boiling water reactor (BWR) nuclear power plant is shown in Figure 5.1. Important parts are reactor, turbines, generator, condenser and valves and pipes connecting the parts.

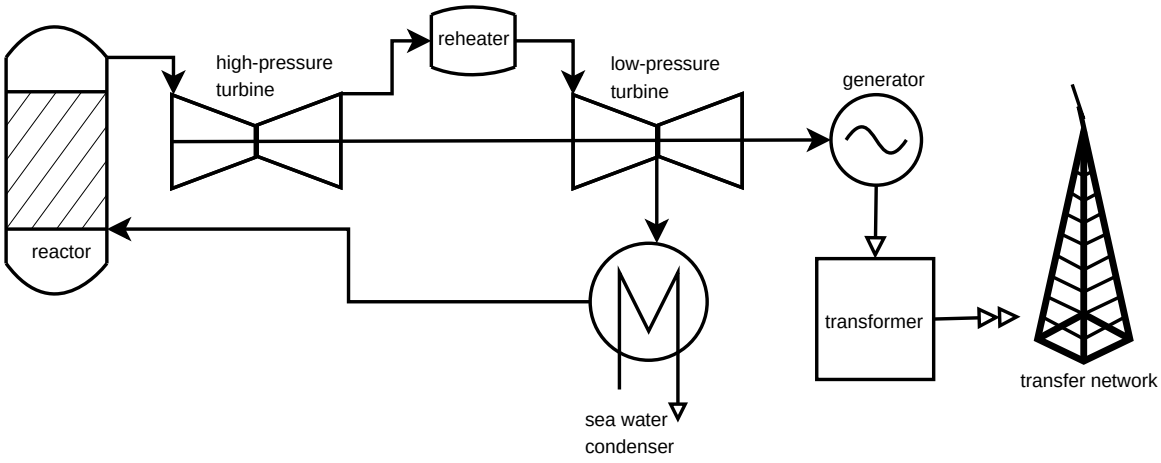

Figure 5.1: Structure diagram of a boiling water reactor (BWR) nuclear power plant.

A nuclear power plant is a safety critical application [10]. A radiation leak could kill a lot of people or make the environment near the plant uninhabitable for hundreds of years. That is why nuclear power plants have especially strict safety requirements. System redundancy is implemented in many ways. For example there are usually four temperature sensors measuring the temperature of one place. If one sensor gets broken, there are still three sensors functional. It is easy to identify a failed sensor by comparing measurements from different sensors. It is highly unlikely that all of the sensors would malfunction simultaneously.

Nuclear power plant systems are build for *defense in depth* (see Section 2.6). The subsystem of the most inner level is the process control system controlling the plant.

Figure 5.2: Pictures of Teollisuuden Voima (TVO) nuclear power plant process operator room.

If it fails the first safety system activates. If also that system fails, a second safety system activates. In the most outer level there is a decision support system (DSS) for mitigation. This system minimizes loss if the worst (reactor meltdown) has already happened. A prototype (RODOS) of such system has been implemented [41].

An operator operates the process control system and some of the safety systems. Control room of the operator is shown in Figure 5.2. The power plant operator has a user interface with control charts, lamps that announce events or state changes, and an *event list* where events appear after some condition becomes true. Inputs of the control system are sensors that are located in different parts of the plant. The sensors measure quantities like pressure, temperature, water height and radiation. Essential actuators of the control system are valves and control rod depth adjusters. The operator has to be able to make decisions in abnormal situations. [1]

5.2 Process of Prototype Development

Software development in the project of this thesis has been highly interactive with requirements changing often so its software process is more similar to agile programming than the waterfall model. The process could be also described as rapid prototyping. The prototype development process is iterative and has been summarized in Figure 5.3. Thinking in terms of *goal*, *inputs* (process variables) and *outputs* (actuators, visualizations) has been a useful tool in finding requirements. The following question is helpful: "What is the goal of the process that is automated or supported by the prototype?".

The *goal* of the prototype is defined as follows: "To provide essential information for a human operator so that he is able to prevent fault states as early as possible and as economically as possible.". *Application domain* is nuclear power plant data in Matlab environment. *Scope* is limited only to the few faults that are in the available datasets. *Inputs* are variables like reactor pressure and temperature. *Outputs* are visualizations like self-organizing map (SOM), time series plots and diagnoses.

Scope, *application domain*, *inputs* and *outputs* have all been evolving during the development process. Especially the amount and quality of *outputs* has increased in

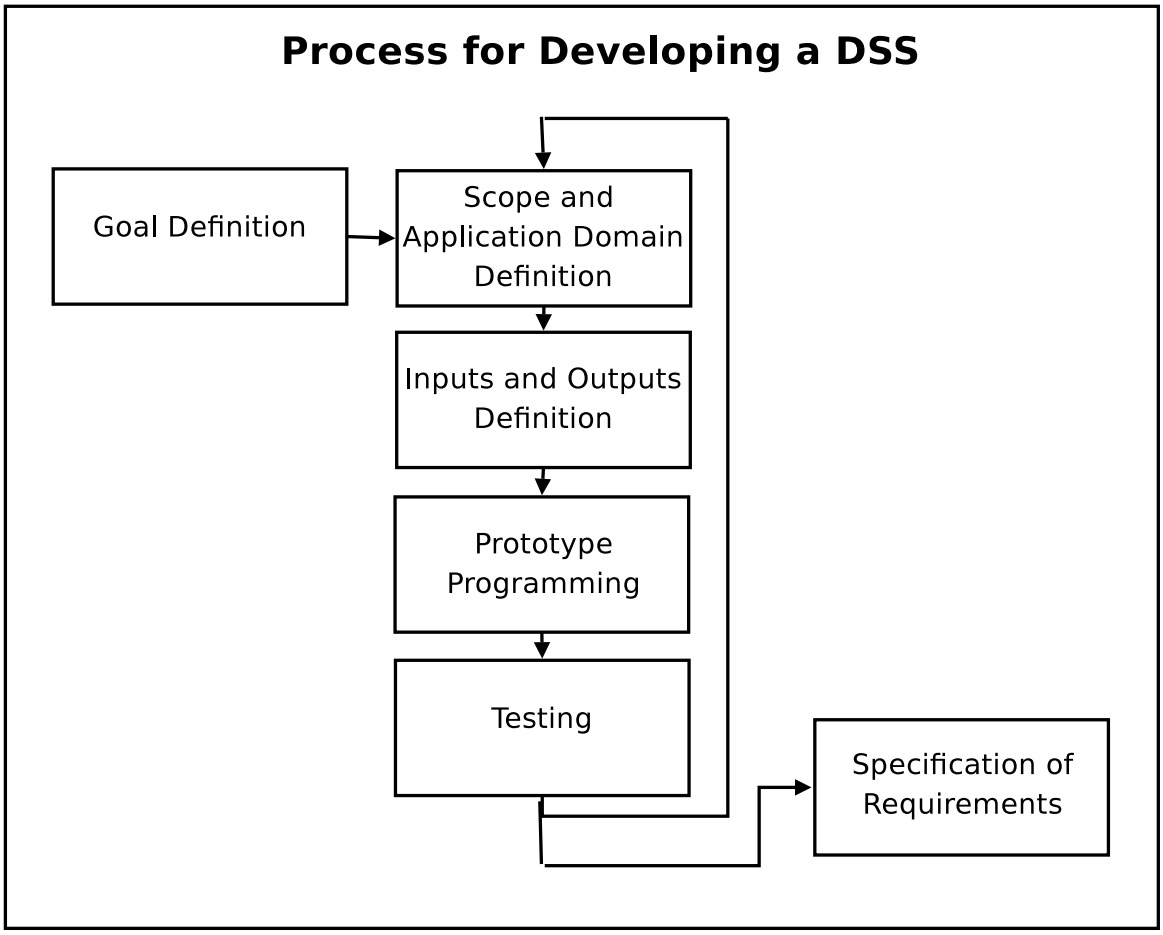

Figure 5.3: Process for developing a DSS. First a goal is defined. After that many iterations of development are conducted. In testing phase when it is clear that the prototype is finished, it is converted into specification of requirements.

every development iteration. The latest prototype version has much better outputs than the first one. After the *specification of requirements* is ready the implementation of a new system begins. This system will be installed into production. Such final specification will probably never be acquired in this project because the nature of its development process is experimental.

A goal hierarchy of decision support systems is illustrated in Figure 5.4. Decision support literature was studied and the discovered goals of decision support belonged to the following group of five goals: prevention, prediction, detection, optimization and human requirements [9, 27, 41, 42, 43, 44, 45]. The goals are part of the goal hierarchy that can be used as a mental model when designing decision support systems.

Data mining has been an essential tool in the prototype development process. It helps to identify which internal structures or knowledge representations would be suitable for the system. In this study the most used data mining methods were SOM and and principal component analysis (PCA). Graphical User Interface (GUI) of the prototype has been designed with intuition and examples from other monitoring and decision support interfaces. Many important factors of GUI design have been left

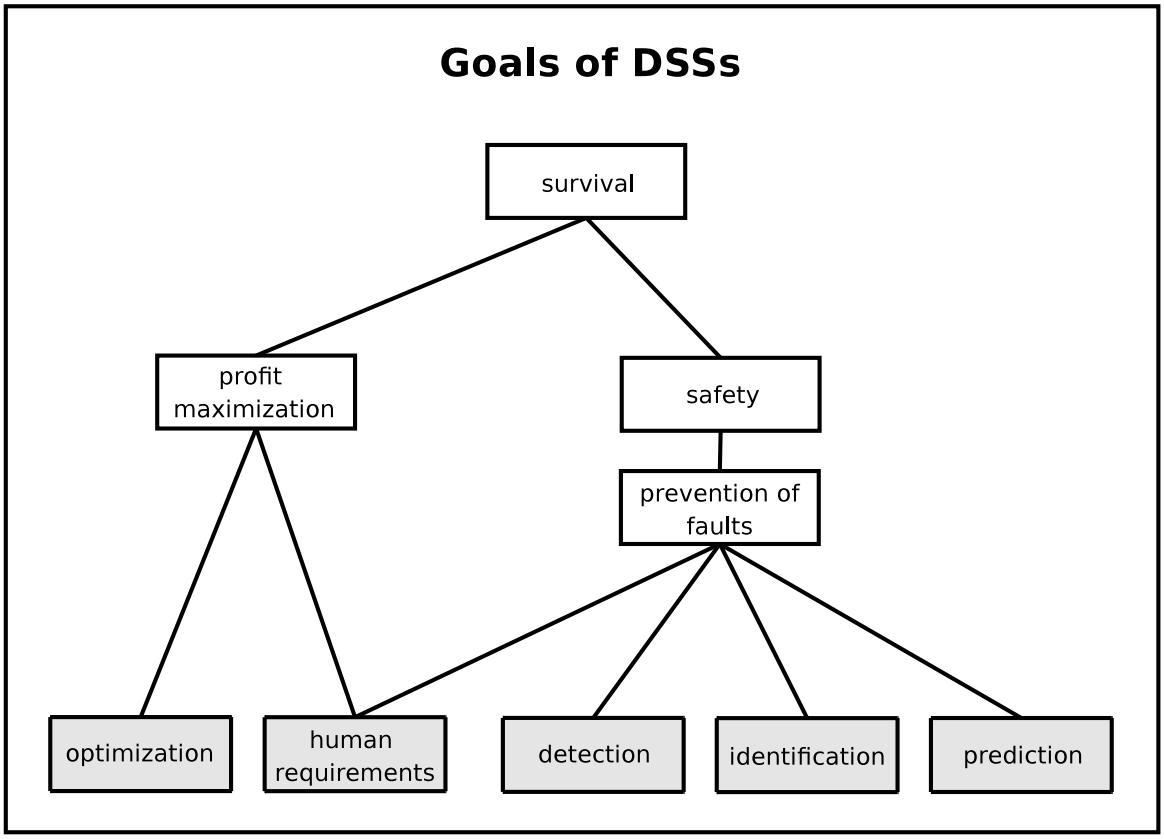

Figure 5.4: Goals of decision support systems organized into a hierarchy.

outside the scope of this study.

5.3 General DSS Model

A general decision support system (DSS) model emerged while developing the prototype. DERSI prototype is one implementation of this model. The model is independent of used data analysis methods and visualizations. Structure of the model is explained in Figure 5.5. Input is processed with data analysis operations and after that used in reasoning diagnoses. The diagnoses are shown in a man-machine interface (MMI) that contains visualizations like plots, text bullets and maps. Those plots are quality control charts associated with warning and alert limits of corresponding variables. Text bullets are descriptions of diagnoses or recommendations. Maps can be anything from SOMs to other projections or diagrams.

A diagnosis can be also used for automatic control. This might be sensible in a serious fault situation. For example the control system of Teollisuuden Voima (TVO) nuclear power plant simulator initiated a reactor scram automatically without any operator interference. The operator should never trust a DSS blindly. He should question the output of the DSS and use also data from other sources as is illustrated in Figure 5.5. Control loop arrow above the DSS box emphasizes that control procedures or operator actions affect the DSS input as a feedback.

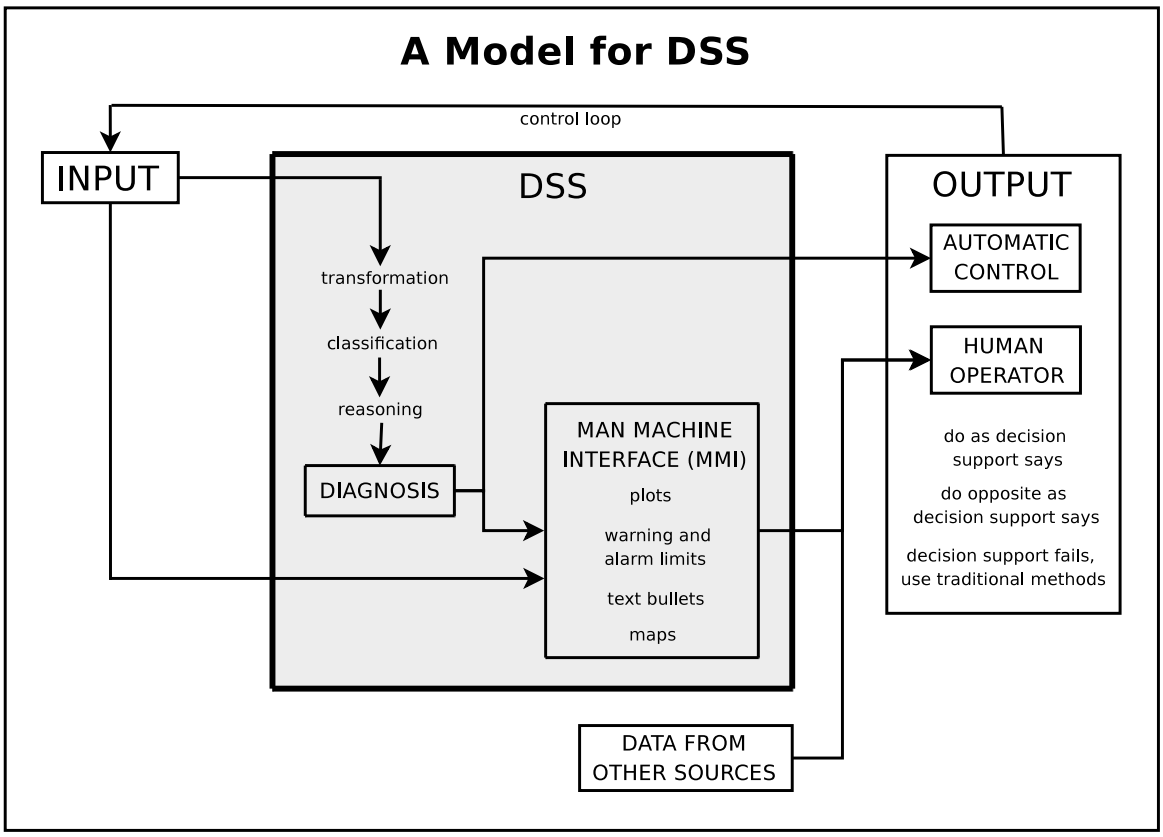

Figure 5.5: General DSS model. DERSI prototype is one implementation of this model.

5.4 DERSI Introduction

DERSI is a DSS platform prototype developed on Matlab programming environment and SOM Toolbox [46] add-on. Its most important property is the usage of SOM quantization error for fuzzy rule-based reasoning. DERSI has a partially object-oriented structure. The prototype is modular and it is easy to extend it with new features. It has been a subject of some earlier publications [47, 48, 49, 50].

DERSI is designed primarily for a nuclear power plant process operator. The original purpose of the prototype was to provide a better operator user interface (UI). During prototype programming it became apparent that appropriate user interface development is beyond the scope of this study. The purpose of the prototype became to demonstrate alternative visualizations and data analysis methods in nuclear power plant decision support. DERSI became an operator UI complement instead of a replacement.

DERSI is a platform that is used for building a DSS (DERSI unit) for an application like nuclear power plant decision support. Datasets of the application and knowledge of its process are needed for the building. The building begins so that Matlab is used for initial data analysis and preparation of the datasets. After that the unit is built one part at a time with the DERSI unit building tools. Dataset matrices are used for teaching SOMs and knowledge of the builder is transformed into rules.

After the unit building is complete the unit can be used for simulation of decision support scenarios.

DERSI has two user interface parts: a GUI and command line tools. The GUI is designed for the supported operator and the command line tools for DERSI unit building. DERSI can also be viewed as a tool collection. Because of its modularity it is easy to add new feature extractors into the prototype.

5.5 DERSI GUI

DERSI GUI with Simulink dataset (see Section 5.8.1) is shown in Figures 5.6 and 5.7. DERSI GUI with TVO simulator dataset (see Section 5.8.2) is shown in Figures 5.8 and 5.9.

The GUI contains 8 frames with different elements. *Frame 1* shows diagnoses and recommendations, *frame 2* shows process state U-matrix and process state U-matrix quantization error, *frame 3* has GUI controls, *frame 4* shows input data plots, sensor quantization error plots and sensor progress value plots. *Frames 5 and 7* show input SOM and sensor SOM component planes and *frames 6 and 8* show bar visualizations. The view of diagnoses and the view of recommendations cannot be shown simultaneously so one or the other has to be selected into their frame. The same applies to the view of input data plots and the view of sensor quantization error and progress value plots. Diagnoses and input data are selected for viewing in Figures 5.6 and 5.8. Recommendations and quantization error plots are selected for viewing in Figures 5.7 and 5.9.

DERSI can be used for simulating scenarios of the DERSI unit application. During such simulation the content of every GUI frame is recalculated in regular intervals (simulation steps).

5.5.1 Diagnosis Frame

Frame 1 is *diagnosis frame* by default. It can be switched to *recommendation frame* by pressing `Diag/Rec` button in *UI Frame*. If many diagnoses are inferred they are listed in priority order from up to down. Each diagnosis has a diagnosis text, a *priority* value *Dpri* in range of 0–100 and a *truthness* value *Dtru* in range of 0–1. Truthness is a fuzzy truth value of the diagnosis.

The process of *Dpri* calculation goes as follows. First for every diagnosis rule k a truth value $Rtru_k$ is calculated by doing fuzzy AND operation for all truth values $Ptru_{k_i}$ of the rule predicates i:

$$Rtru_k = \min_i \{Ptru_{k_i}\} \tag{5.1}$$

Predicates are explained in in detail in Section 5.6. Fuzzy OR is not supported in DERSI rules because it can be implemented by negations and AND operations [39]. After rule truth calculations it is possible that a diagnosis l gets many truth values from different rules j. Diagnosis *truthness* of that diagnosis is calculated by choosing the maximum value of these:

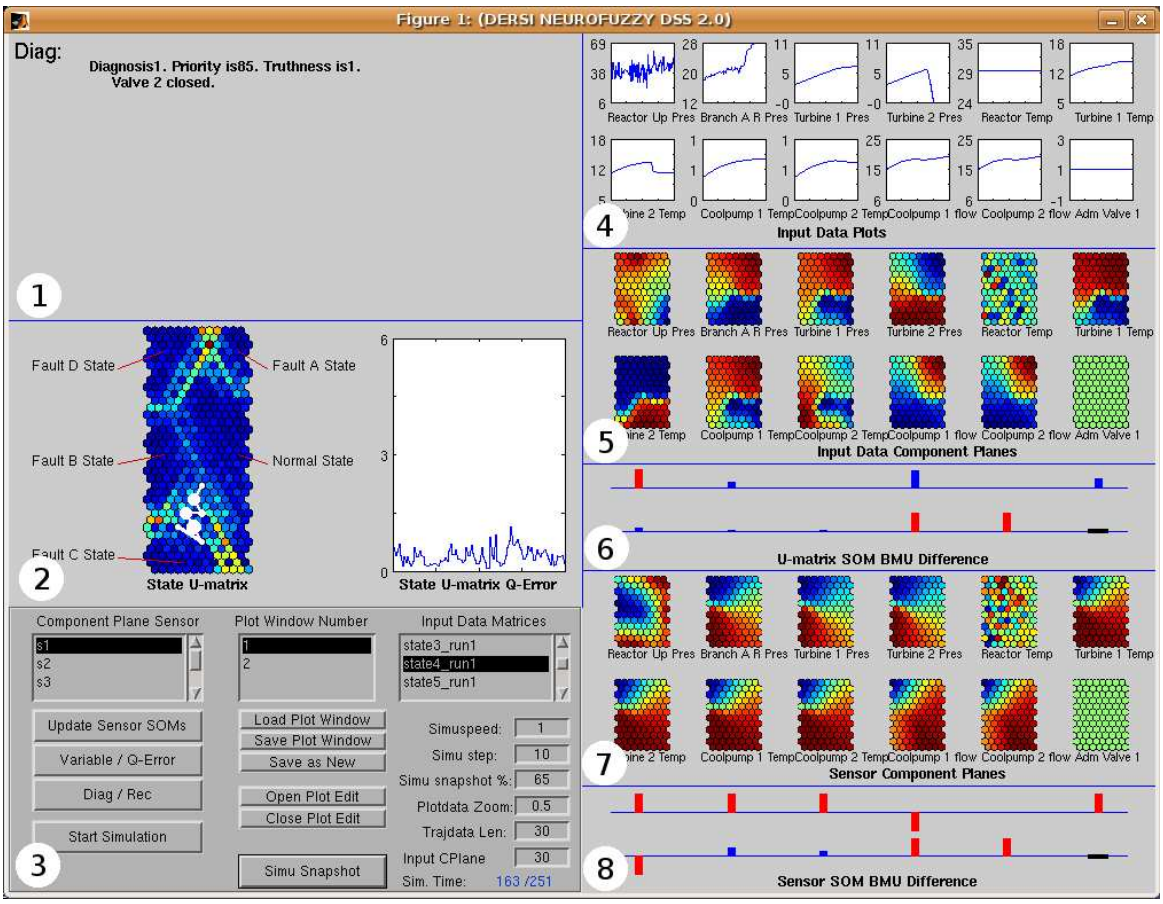

Figure 5.6: DERSI GUI of Simulink unit with input data plots in *frame 4* and diagnoses in *frame 1*.

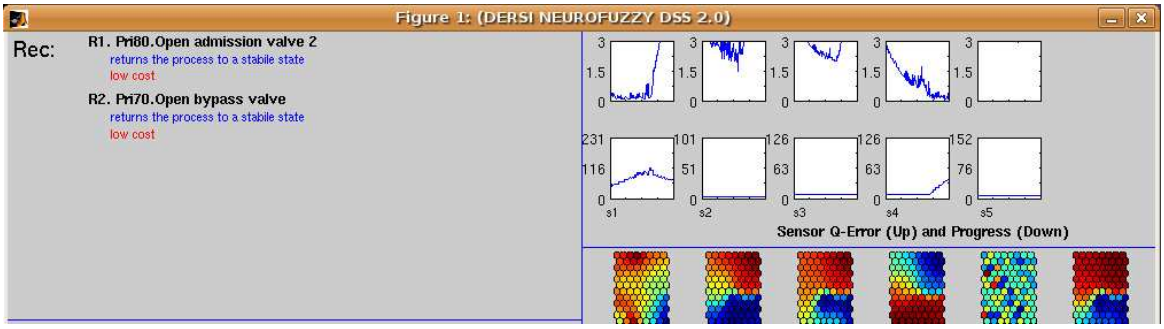

Figure 5.7: Upper frames of DERSI GUI of Simulink unit. Quantization error and progress value plots are shown instead of the input data plots in *frame 4* and recommendations are shown instead of diagnoses in *frame 1*.

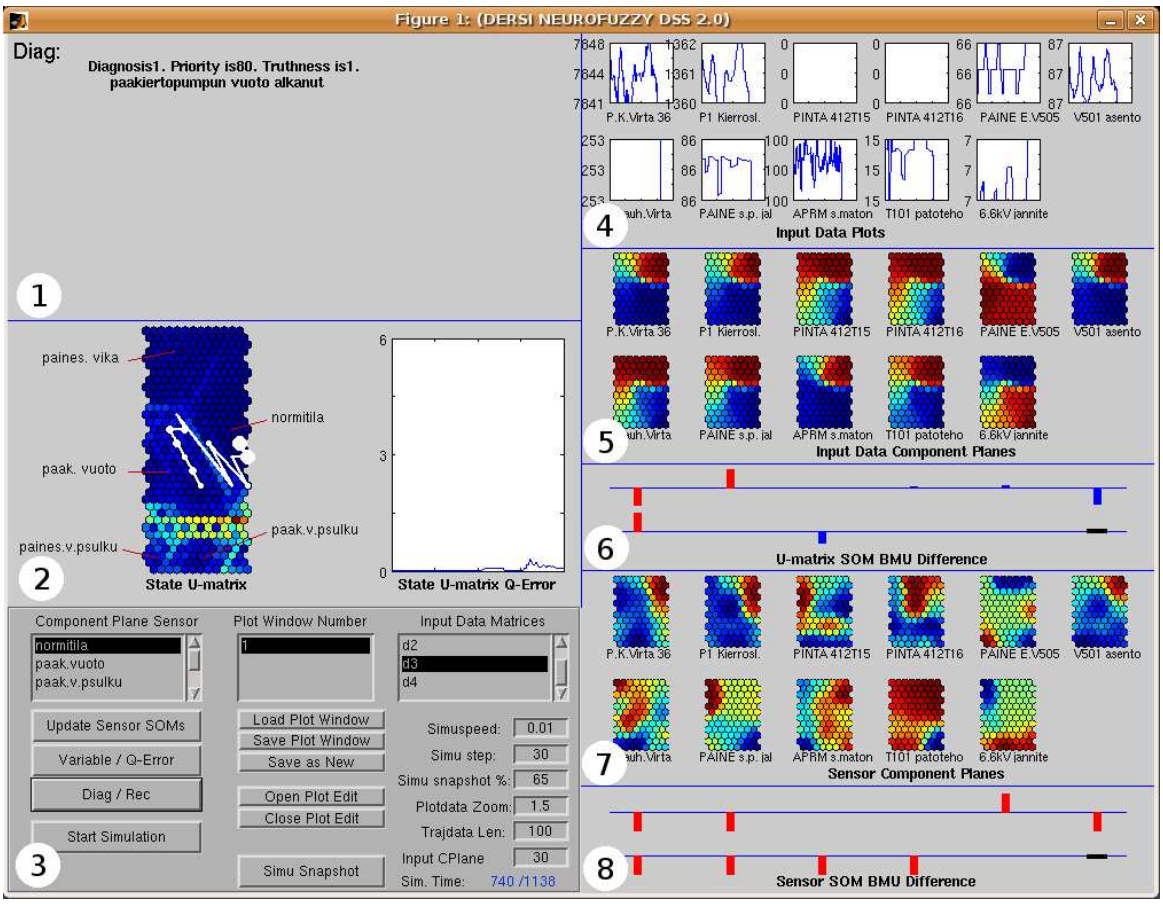

Figure 5.8: DERSI GUI of TVO unit with input data plots in *frame 4* and diagnoses in *frame 1*. Some texts are in finnish language.

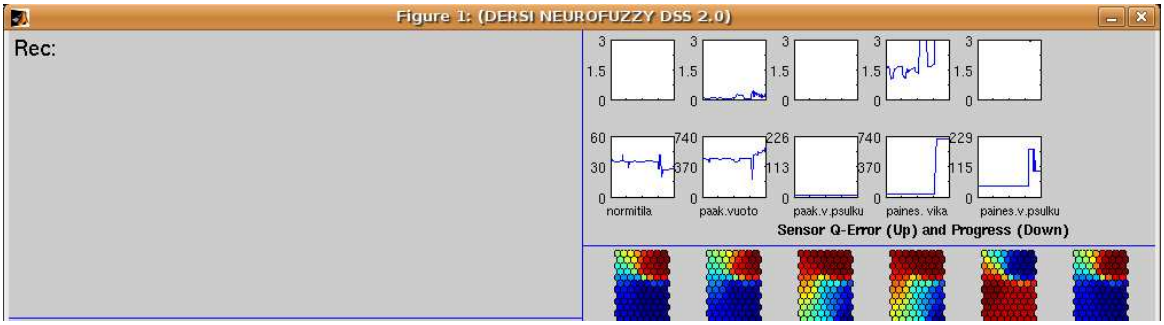

Figure 5.9: Upper frames of DERSI GUI of TVO unit. Quantization error and progress value plots are shown instead of the input data plots in *frame 4*. Some texts are in finnish language.

$$Dtru_l = \max_j \{Rtru_{l_j}\} \tag{5.2}$$

Choice of maximum value makes sense because false positive diagnosis is much cheaper than false negative diagnosis in the nuclear power plant application that DERSI is designed for. If the diagnosis gets a truth value from only one rule, that value will be the diagnosis *truthness*. The *priority* of diagnosis l can be calculated by multiplying its diagnosis *truthness* and *significance*:

$$Dpri_l = Dtru_l \cdot Dsig_l \tag{5.3}$$

Significance is a part of a diagnosis entity and it means the importance of a particular diagnosis. The diagnosis significance is constant unlike the diagnosis priority. If the *priority* gets a value that is less than a specified *threshold value*, the diagnosis of that priority will not be shown in the frame.

5.5.2 Recommendation Frame

Frame 1 can be switched to the *recommendation frame* by pressing the `Diag/Rec` button in *UI Frame*. Recommendations are also shown in priority order from up to down. They have own fields for the descriptions of *effect* and *cost* of the recommended action.

The process of recommendation *priority* value $RECpri$ calculation goes as follows. First the truth values of recommendation rules are calculated in a similar way as the truth values of diagnosis rules. For every recommendation rule k a truth value $Rtru_k$ is calculated by doing fuzzy AND operation for all truth values $Ptru_{k_i}$ of the rule predicates i:

$$Rtru_k = \min_i \{Ptru_{k_i}\} \tag{5.4}$$

$RECpri$ values are calculated differently than diagnosis priorities $Dpri$. One recommendation rule can have many recommendations and their significances as a consequent. A recommendation *significance* is a part of rule entity and not a part of recommendation entity, so a particular recommendation can have different significances in different rules. If many rules produce truth values for the same recommendation, the higher significance rules should have more weight in *priority* calculation than the lower significance rules. The recommendation priority $RECpri_l$ is calculated with weighting accomplished by multiplication of rule truth values $Rtru_{l_j}$ and significances $RECsig_{l_j}$:

$$RECpri_l = \max_j \{Rtru_{l_j} \cdot RECsig_{l_j}\} \tag{5.5}$$

l is the index of a recommendation and j is the index of a rule having recommendation l in its consequent.

5.5.3 Input Data Plot Frame

Frame 4 is *input data plot frame* by default. It can be switched to *sensor q-error plot frame* by pressing the `Variable/Q-Error` button in the *UI Frame*. Input data comes from processes of simulation scenarios which are listed in `Input Data Matrices` field of *UI Frame*.

Input data plots are similar to quality control charts of process control. They show process variable values. The most recent values are in the right side of the plot rectangles and the oldest values are in the left side. Variable names are written below the rectangles. Reactor pressure is an example of a process variable. The plot y-axis scales are specified in the DERSI unit building phase and they are based on the means and variances of the teaching dataset variables. The x-axis scales can be adjusted by writing a new value in the `Plotdata Zoom` field of the *UI Frame*. This changes the scales of all plots of the DERSI GUI.

Variables 1–12 are shown in the *input data plot frame* by default. If the DERSI unit has more than 12 process variables, a view with different variables can be observed by first choosing a new view from the `Plot Window Number` list of *UI frame* and then pressing the `Load Plot Window` button below the list. A variable view can be customized in the following way. First the `Open Plot Edit` button is pressed, then new variables are chosen from the appearing menus. After that the `Close Plot Edit` button is pressed. Finally either the view chosen in the `Plot Window Number` list is overwritten by pressing the `Save Plot Window` button or a new view is created by pressing the `Save as New` button.

5.5.4 Sensor Q-Error Plot Frame

Frame 4 can be switched to the *sensor q-error plot frame* by pressing the `Variable/Q-Error` button in the *UI Frame*. Sensor SOM quantization errors are plotted in the upper row and sensor SOM progress values in the lower row of the frame. Names of the sensors are written below the progress value plots. Sensor plot views can be customized and plot x-axis scales adjusted in the same way as views and scales of input data plots.

A sensor refers to an object of the `somsensor` class whose details are explained later in the text. The purpose of the sensor is to represent and identify the state of the process of the DERSI unit simulation scenario. This state has been taught for the SOMs of the sensor in the DERSI unit building phase. One way for the identification is to plot the SOM quantization error of the input data on a sensor SOM. The quantization error plots can be used for monitoring the process. It is likely in a state represented by a sensor whose latest quantization error is low. It is probably not in a state represented by a sensor whose latest quantization error is high.

Sensor X *SOM progress* is the time value of the sensor X teaching data vector that is similar to the vector of the current state. If a fault state is identified with a SOM quantization error, the SOM progress can then be used for identifying the phase of the fault, although this works only in limited amount of situations. Before the calculations of SOM quantization and progress values the input data has to be transformed into the `somsensor` space. The transformation is explained in Section

5.6.

5.5.5 Process State U-Matrix Frame

Frame 2 is the *process state U-matrix frame*. A SOM U-matrix visualization is shown in the left side of this frame. The U-matrix of the visualization is contained in a `umatrix` object whose details are explained later in the text. Process states are mapped to different areas or clusters in this state U-matrix. Clusters are shown as darker areas and cluster borders as lighter areas. Clusters are labeled with texts on the left and right sides of the U-matrix. A trajectory is drawn into the U-matrix and one end of the trajectory represents the current state of the process. Other points of the trajectory are past process states. The opposite end of the trajectory is X samples in the past where X is defined by a value of `Trajdata Len` field in the *UI Frame*. Process state transitions can be monitored by watching how the trajectory moves between the different clusters.

In the right side of the frame is the state U-matrix quantization error plot. Input data quantization errors on the state U-matrix SOM are visualized with this plot. The plot can be used for two purposes. First when the trajectory crosses a cluster border, a peak is also shown in this plot and it is easy to verify state transitions. Second if the process enters a previously unknown state, the U-matrix quantization error will get very large and it is easy to detect such unknown process states. Before the calculations of the U-matrix trajectory and the state U-matrix quantization error plot the input data has to be transformed into the `umatrix` space. The transformation is explained in Section 5.6.

5.5.6 Input Data SOM Component Plane Frame

Frame 5 is the *input data SOM component plane frame*. The component planes can be used for monitoring correlations between different process variables of the input data. The plane visualization is created by first forming a matrix from X most recent vectors of the current scenario data where X is `Input CPlane` field value of *UI Frame*. A SOM with fixed size and topology is then constructed from this matrix and the SOM component planes are visualized in the frame.

5.5.7 Sensor Data SOM Component Plane Frame

Frame 7 is the *sensor data SOM component plane frame*. Every sensor has a SOM dedicated for the component plane visualization. The currently visible planes can be switched to those of a different sensor by first choosing a new sensor from the `Component Plane Sensor` list of the *UI Frame* and then pressing the `Update Sensor SOMs` button. The component planes can be used for checking correlations between different process variables in the known process states whose data have been used for teaching the sensors. Found correlations can be then compared with correlations of the input data that are visible in *Frame 5*.

Sensor SOM of component plane visualization is different than that of quantization error calculation because the size and topology of the plane visualization SOM has to

be fixed. Without using the same SOM size and topology in both plane visualization frames it would be difficult to compare the correlations between these two frames.

5.5.8 U-Matrix Bar Visualizations Frame

Frame 6 is the *U-matrix bar visualizations frame*. The bars are formed in the following way. The latest input data vector is first transformed into the `umatrix` space. Then the difference between the transformed vector and its BMU in the process state U-matrix is calculated. The difference is then inverse transformed back into the original space and visualized with blue vertical bars. If a component value of the difference vector grows over a certain threshold, color of the corresponding bar changes into red and the height of the bar stops growing. This is because the GUI height is limited. If the component gets a NaN (not a number) value, it is visualized with a wide and shallow black bar. The frame reveals which variables are the ones responsible for a high state U-matrix quantization error.

5.5.9 Sensor Bar Visualizations Frame

Frame 8 is the *sensor bar visualizations frame*. These bars are similar to those of the *U-matrix bar visualizations frame* but the SOM used in the BMU calculation is that of the currently chosen sensor that is highlighted in the `Component Plane Sensor` list of *UI Frame*. The frame reveals which variables are the ones responsible for a high sensor SOM quantization error of the chosen sensor.

5.5.10 UI Frame

Frame 3 is the *UI frame*. `Input Data Matrices` field has a list of available process scenarios. A scenario simulation can be started by first choosing a scenario from this list and after that pressing the `Start Simulation` button. The prototype is quite slow, so calculation of a new simulation instant can take sometimes up to 10 seconds. Current simulation instant is shown as the left number of `Sim. Time:` field in the lower right corner of the frame. The right number is the length of the simulation. Simulation speed can be adjusted by writing a new value to the `Simuspeed:` field. Simulation runs in discrete steps. `Simu step:` field value defines how many samples are jumped forward in each step of the simulation.

It can be convenient to take a snapshot of a specific instant of the simulation. First the desired snapshot instant is written in `Simu snapshot %:` field (0 = beginning, 100 = end). Then a snapshot can be generated by pressing the `Simu Snapshot` button.

5.6 DERSI Structure

In DERSI, the most important decision support knowledge is saved in five types of entities: SOM sensors, predicates, rules, diagnoses and recommendations. Figure 5.10 shows important DERSI entities and their relations. The DERSI prototype unit is a Matlab struct, that is normally saved in a Matlab global variable named `DECS`. `inputData` contains all input data matrices (scenarios) of the process. `dssgui`

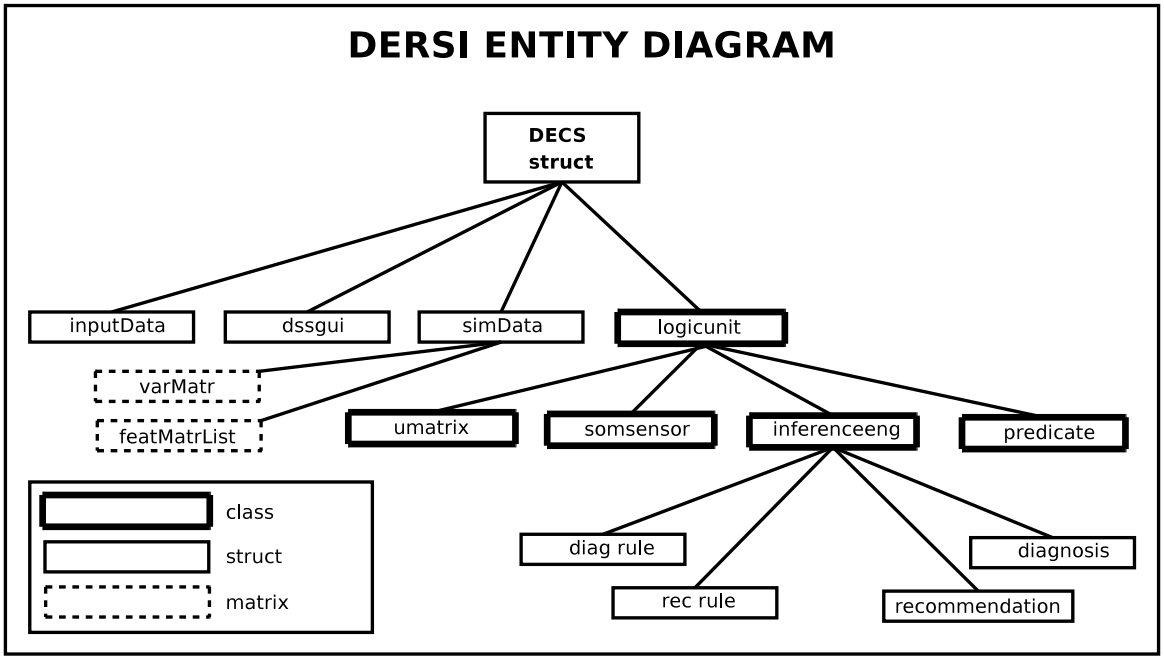

Figure 5.10: DERSI entity diagram. The most important entities of DERSI platform and their relations are shown.

Original Matrix

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18

Matrix With Delayed Variables

org	*org*	*org*	*d1*	*d1*	*d1*	*d2*	*d2*	*d2*
1	2	3	NaN	NaN	NaN	NaN	NaN	NaN
4	5	6	NaN	NaN	NaN	NaN	NaN	NaN
7	8	9	1	2	3	NaN	NaN	NaN
10	11	12	4	5	6	NaN	NaN	NaN
13	14	15	7	8	9	1	2	3
16	17	18	10	11	12	4	5	6

Figure 5.11: Time information expansion of a data matrix. Time information is added to a data matrix with delayed copies of its columns. Amount of delay b is 2 samples and amount of "echoes" $a-1$ is 2. NaN means missing variable values.

contains the most important program variables that are associated with the state of DERSI GUI components. `simData` contains the state of the DSS scenario simulation, most importantly `varMatr` that is a matrix and `featMatrList` that is a cell array of matrices. `varMatr` contains all process variables like *temperature* and *pressure* in its columns. Rows of `varMatr` represent different instants of time. First row of the matrix is the latest instant t, second row is the next latest instant $t-1$, row N is the

Nth latest instant $t - N$. Columns of `featMatrList` matrices are features extracted from process variables. The first `featMatrList` matrix gets `somsensor` quantization errors and the second gets `somsensor` progress values. The matrices have as many rows as `varMatr` and equal rows in both represent equal time instants.

`logicunit`, `umatrix`, `somsensor`, `inferenceeng` and `predicate` are the classes of the prototype. `logicunit` is a container of all DSS logic. It contains a `umatrix`, `somsensor`s, an `inferenceeng` and `predicate`s. `somsensor` contains three SOMs. One is for quantization error calculation, one is for component plane visualization, and one is for SOM progress value calculation. `umatrix` contains one SOM, and the U-matrix of that SOM. The SOM of the `umatrix` is created so that teaching data of all `somsensor`s are combined into one big teaching data matrix. It is the transformed into the `umatrix` space and used in the teaching of the `umatrix` SOM.

Before all calculations of `somsensor` or `umatrix` methods, used input data is transformed to the corresponding `somsensor` or `umatrix` space. Transformation parameters are attributes of objects of these classes. The transformation happens as follows. First the input data matrix is normalized into *zero mean unit variance* of the teaching data of the class object. Then a subset S of the variables is selected and their columns are copied into a temporary matrix. This temporary matrix is then processed. Time information is created by making $a-1$ delayed copies of every column so that they are delayed with integer multiple of b as shown in Figure 5.11, where $a = 3$ and $b = 2$. Parameters S, a and b have been decided when building the DERSI unit. These columns are then combined with the temporary matrix into the final transformed matrix that is then used for the `somsensor` and `umatrix` calculations. Some calculation results have to be inverse transformed before further utilization. The SOM algorithm forgets the time relations of teaching samples and this is the main reason for providing DERSI platform with the ability to include delayed copies of variables in the SOM teaching and calculations.

`predicate` is a fuzzy logic predicate, see Figure 5.12. It contains a *membership function* that is generally trapezoid shaped, as shown in *field 1* of the Figure. Shape of the trapezoid is defined by the values *maybe_down*, *is_down*, *is_up* and *maybe_up*. The `predicate` is associated with some process variable (for example *reactor pressure*), that is in the x-axis of the predicate membership function chart. y-axis of the chart is the truth value of the predicate. The predicate truth for a value X of a process variable can be found by checking which value does the membership function get at point X. In DERSI it is possible to have a membership function whose other edge is at positive or negative infinity, as shown in *fields 2, 3, 5* and *6* of the Figure. Membership functions of propositional logic predicates $A < x < B$, $x > A$ and $x < B$ are shown in *fields 4, 5* and *6*.

`inferenceeng` is the prototype inference engine and knowledge base. It contains four lists: 1. `diag rule` list (rule consequent is a `diagnosis`), 2. `rec rule` list (rule consequent is a set of `recommendation`s), 3. list of `diagnosis` and 4. list of `recommendation`s. `diagnosis` is a struct that contains a *problem description* in text form and a diagnosis *significance*. `recommendation` is a struct that contains a *recommended action*, *cost* of the action and *effect* of the action as text descriptions. Every `rec rule` contain separate *significances* for the `recommendation`s of the rule. So a particular `recommendation` can have different significances in different rules. An

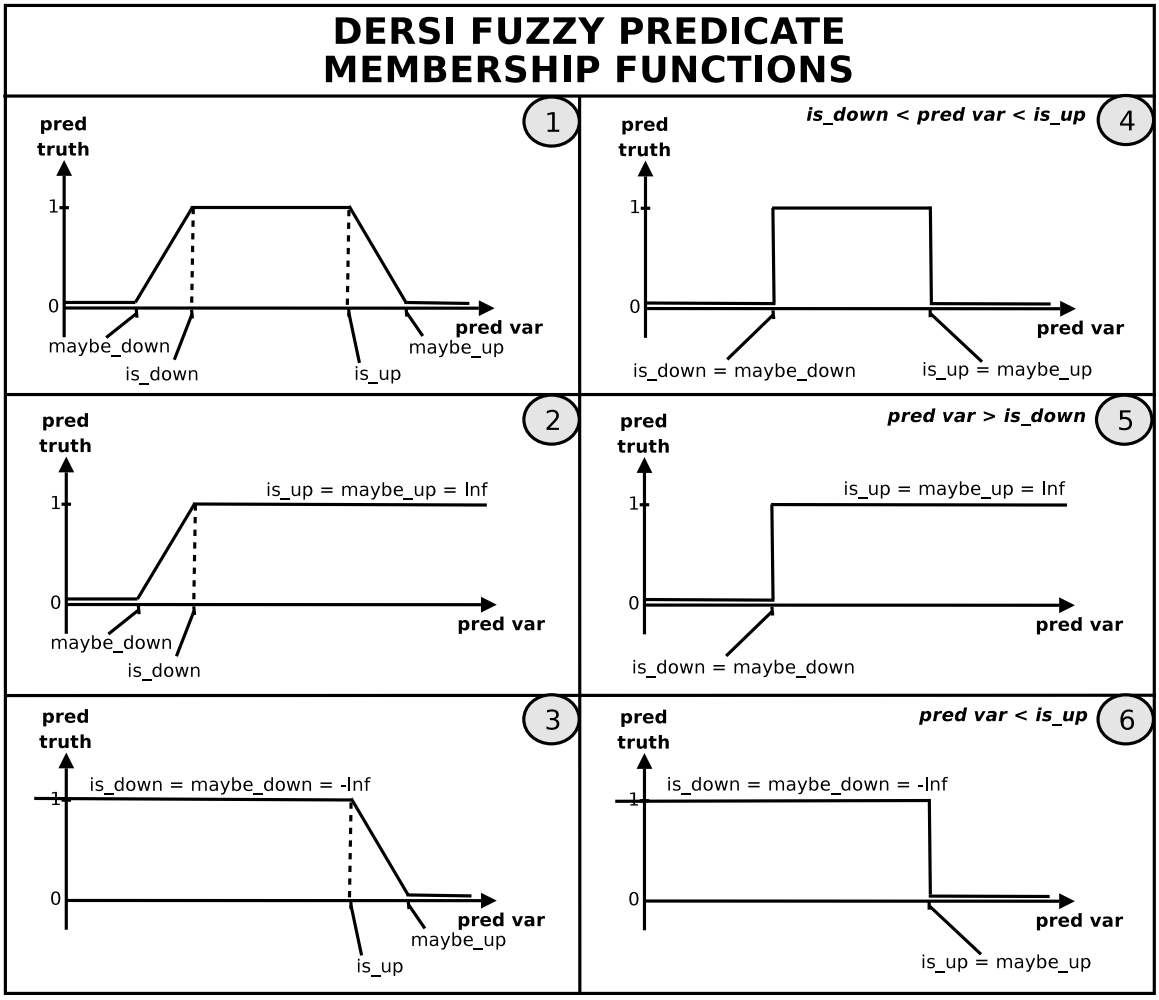

Figure 5.12: DERSI fuzzy predicate membership functions. Fields 1, 2 and 3 are fuzzy membership functions and fields 4, 5 and 6 are corresponding propositional logic membership functions.

example of a diagnosis rule and a recommendation rule are shown in Figure 5.13. They are from DERSI Simulink unit that is explained in Section 5.8.1.

The concept of *diagnosis significance* is absolute. For example the diagnosis *reactor meltdown* is always a very serious condition. But a breakage of one of four sensors measuring the same process quantity is a smaller problem. The concept of *recommendation significance* is context dependent. For example it would not make sense to use boron injection to fix the problem of a temperature sensor because boron injection is an expensive action. In this situation it would be better to set a small significance value for boron injection. But if a *reactor meltdown* is diagnosed it would make sense to set a high significance value for the boron injection.

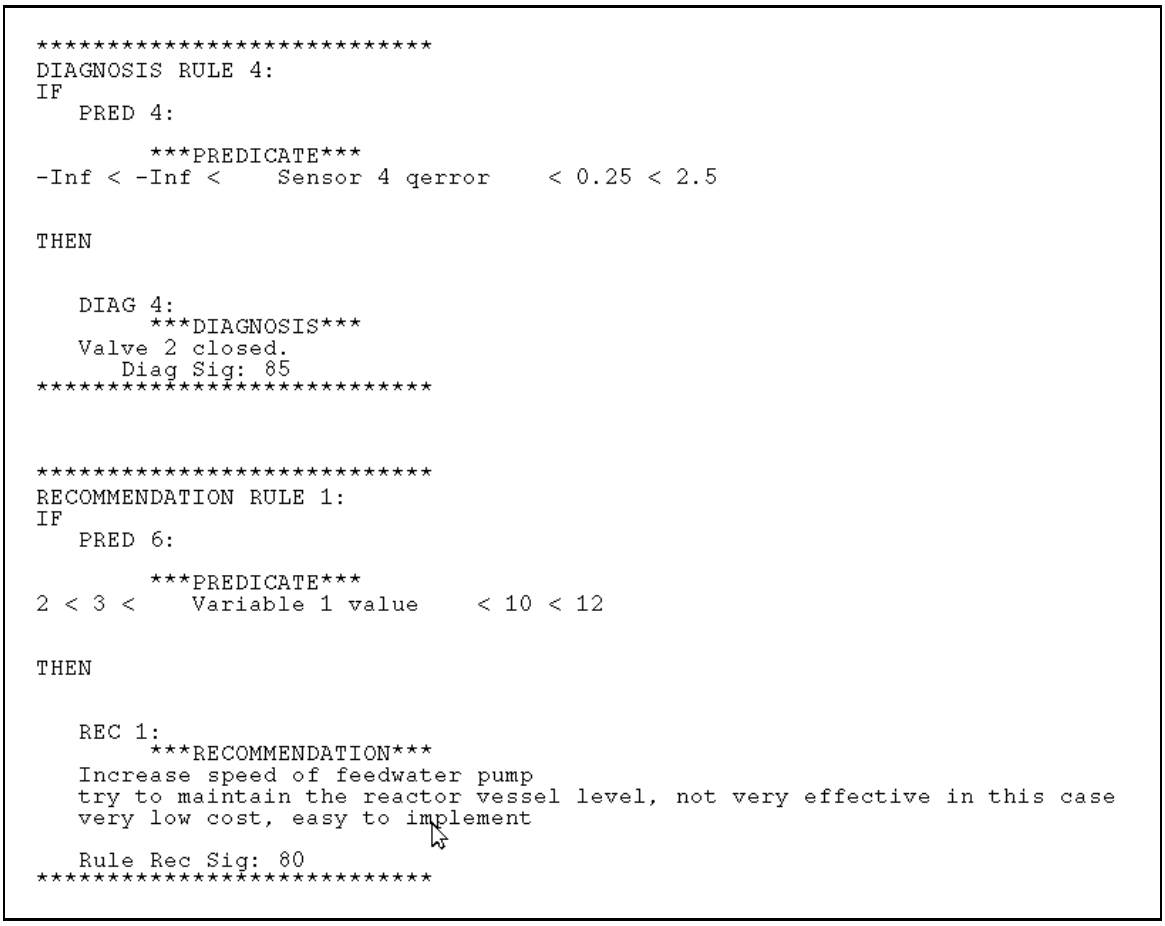

Figure 5.13: DERSI diagnosis rule (above) and recommendation rule (below). Words predicate and significance are abbreviated as pred. and sig.

5.7 DERSI Operation

Important interactions between the DERSI entities are shown in Figure 5.14. The most important prototype functionality is in two files: `modify_dss.m` contains functionality for building and modifying a DERSI prototype unit and `dssgui.m` contains GUI drawing functions, GUI callback functions and simulation logic. Almost all function calls in DERSI operation or unit building are initiated from one of these files. `logicunit` interacts with `modify_dss.m`, `dssgui.m` and class objects that are contained in it. `inferenceeng` interacts with `logicunit` and data that is contained in it. Only three entities interact with `SOM Toolbox`: `umatrix` and `somsensor` as they contain SOMs, and `dssgui.m` as it uses `SOM Toolbox` functions for visualizing the SOMs.

DERSI operation in simplified form is shown in Figure 5.15 and DERSI inference process is illustrated in Figure 5.16. At every simulation step `varMatr` is updated with latest simulation samples from an input data matrix. Then `varMatr` is used for calculating features and copying them to matrices in `featMatrList`. In the current DERSI version only SOM quantization errors and progress values are used as extracted

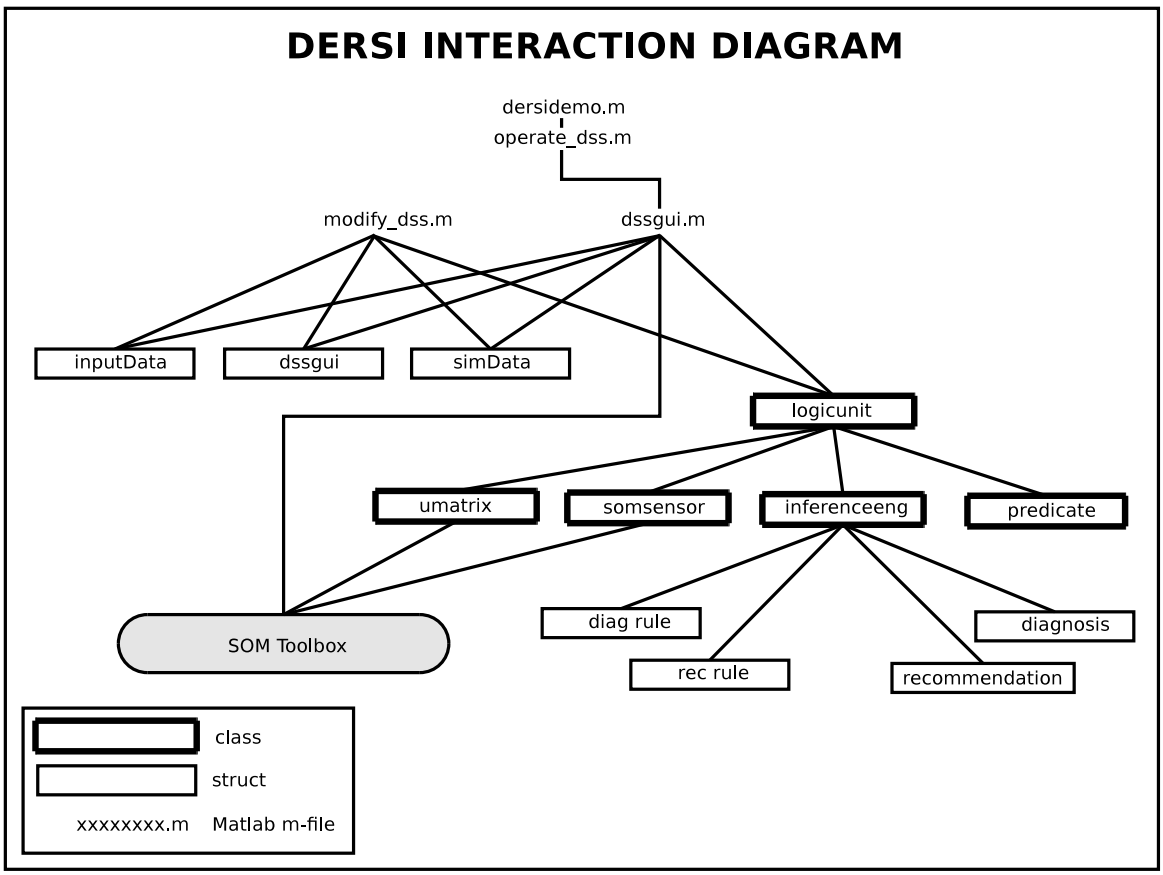

Figure 5.14: DERSI interaction diagram. The most important interactions between DERSI entities are visualized with lines between them.

features. After this inference results and visualizations are calculated and finally drawn into the GUI.

Inference process shown in Figure 5.16 is explained as follows. The most recent (instant t) values of process variables and calculated features are used as an input in *fuzzification 1*. Acquired predicate truth values are used as an input in *rule inference 1*, that produces diagnoses with priorities and truth values (truthness). If the rule base has rules that use diagnosis truth values as an input, the values are fuzzified with predicates of those rules (*fuzzification 2*). Acquired diagnosis predicate truth values are used together with truth values of *fuzzification 1* for a second inference (*rule inference 2*). The second inference is needed because sometimes it makes sense to infer recommendations from found diagnoses.

5.8 DERSI Unit Building

As DERSI is a *DSS platform*, building a DSS unit (*DERSI unit*) for some application is a separate process. `modify_dss.m` is the most important Matlab function that is used for building a prototype unit. It is a command line tool and its text interface is shown in Figure 5.17.

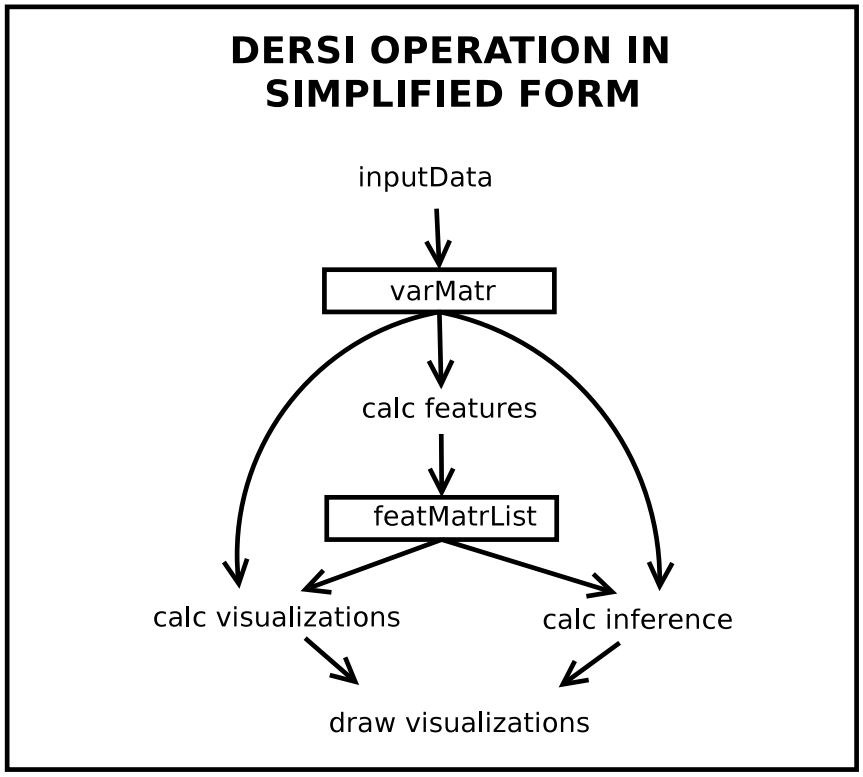

Figure 5.15: DERSI operation in simplified form.

The nature of the process datasets need to be known when creating SOM sensors, rules and other prototype entities. Data mining is a part of DERSI unit building and it can be done both with DERSI platform and external tools. Choice of teaching samples and variables is significant when teaching DERSI SOMs. If the application has very large amount of variables, the dimension has to be reduced even before starting to build the DERSI unit. Amount of time information (delayed variable copies) that is to be included in SOM teaching has to be decided.

As *classification* with quantization errors is one of the main principles of DERSI, the used application data has to be always divided to separate data matrices that represent different classes. For example it is possible that data from a process with fault is provided in a form of one long time series. In the beginning of the series the process is in a *normal state* and at some instant a *fault* happens. In such case a part of the time series would need to be copied into one matrix and a part into another.

If the fault is slowly developing, it is a difficult problem to decide which portion of the time series is effective to use for a fault classifier building. t_{fault} refers to the sample index of the instant when the fault starts. As the goal of the decision support is usually early fault identification, it makes sense to use the samples after t_{fault}.

The classifiers need also rules. Fuzzy membership functions need to be set separately for every rule so that they activate at desired instants. It is obviously impossible to create a classifier for unknown fault states. Such faults can be detected by creating a rule that has *q-error* > *limit* type predicates of all known process state classifiers as an antecedent. Such faults can be also detected with U-matrix SOM quantization

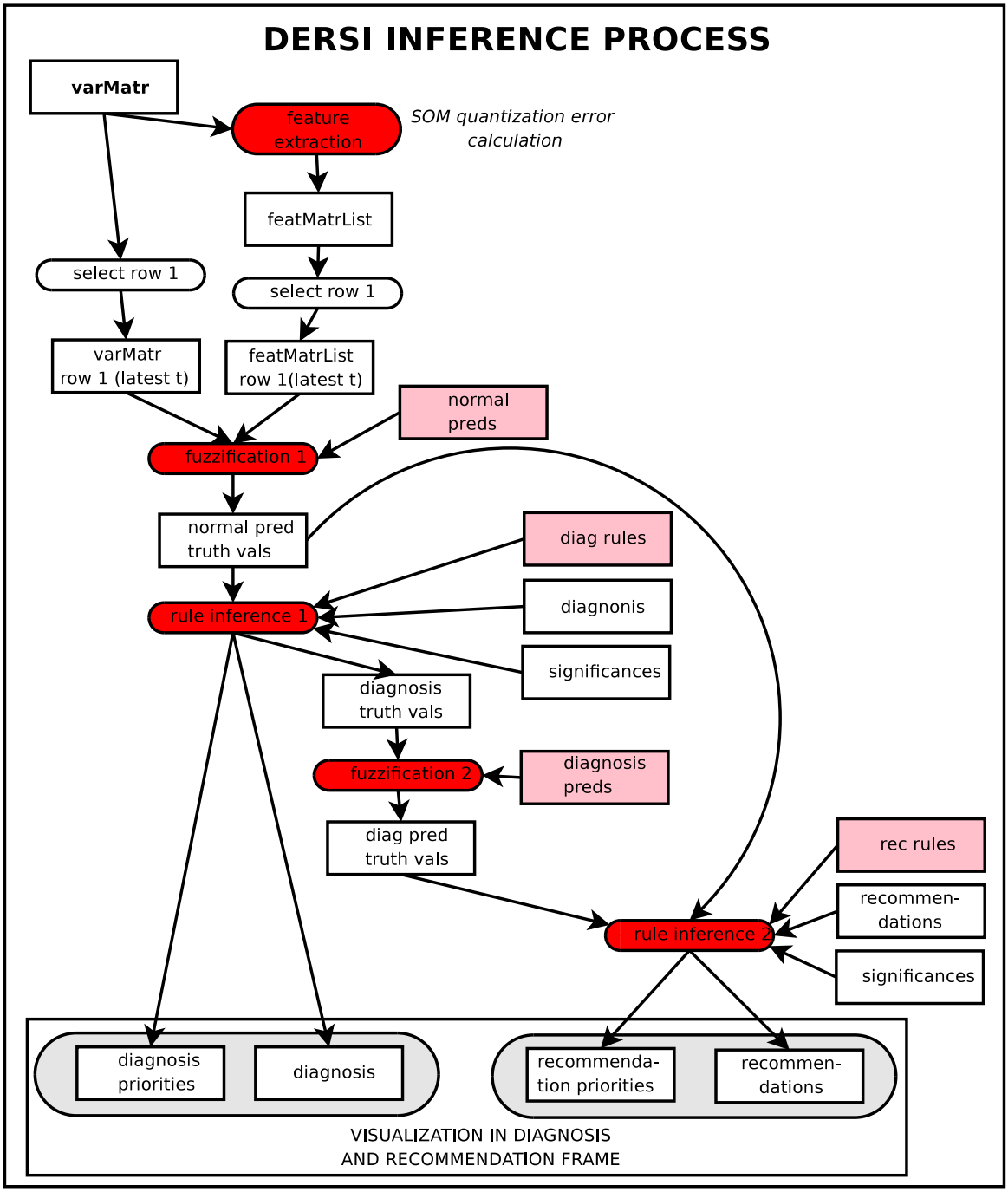

Figure 5.16: DERSI inference process. Words predicate and value are abbreviated as pred. and val.

error plot as stated in Section 5.5.5.

The rule base should be as complete as possible [39]. A complete rule base has rules for handling all possible inputs and conditions. It is impossible to prove a rule base to be complete in a nuclear power plant application because there is always the likelihood of a previously unidentified fault happening. But it should be made

```
            ***   MODIFY_DSS HELP   ***

Create Sensor:              modify_dss(21, sensorNumber, matrix)
Erase Sensor:               modify_dss(29, sensorNumber)

Create Normal Predicate:    modify_dss(31, predNumber)
Create Diagnosis Predicate: modify_dss(32, predNumber)
Erase Diagnosis Predicate:  modify_dss(38, predNumber)
Erase Normal Predicate:     modify_dss(39, predNumber)

Create Diagnosis Rule:      modify_dss(41, ruleNumber)
Create Recommendation Rule: modify_dss(42, ruleNumber)
Erase Recommendation Rule:  modify_dss(48, ruleNumber)
Erase Diagnosis Rule:       modify_dss(49, ruleNumber)


Create Diagnosis:           modify_dss(51, diagNumber)
Create Recommendation:      modify_dss(52, recNumber)
Erase Recommendation:       modify_dss(58, recNumber))
Erase Diagnosis:            modify_dss(59, diagNumber)

Setup  U-Matrix Parameters: modify_dss(61)
Create U-Matrix:            modify_dss(62)
Create U-Matrix Label:      modify_dss(63, labNumber)
Erase U-Matrix Label:       modify_dss(64, labNumber)

Create Input Data Set:      modify_dss(71, setNumber, matrix, setName)
Erase Input Data Set:       modify_dss(79, setNumber)


Create Variable Names:      modify_dss(3, lowVar, highVar)
Create / Reset Plot Wins:   modify_dss(4)

>>
```

Figure 5.17: `modify_dss.m` user interface. It is used for DERSI unit building.

certain that all essential knowledge is coded into rules. Generation of a complete fuzzy rule base is a large problem and was left outside the scope of this study. Intuition, experimentation, trial and error were used in building classifiers and the rule base.

5.8.1 DERSI Simulink Unit

A Simulink model of a boiling water reactor (BWR) nuclear power plant process was built in this study. The model is simplified and based largely on assumptions. Its components are modelled in total with 71 differential equations. PI-diagram of the model is shown in Figure 5.18. An illustrative demonstration of DERSI was acquired with a DERSI unit that is constructed from a dataset of this Simulink model. The rules of this DERSI Simulink unit can be found from Appendix A. Details of the model are in a report of a special study [51].

Six different scenarios are simulated with the process model. All of the scenarios begin with a start of the nuclear reactor. One scenario is a *normal state* and the rest are *fault states*. In scenario 1 starting of the reactor is successful and the process converges into a stabile normal state. In scenario 2 a leakage appears between `Reactor` and `Preheater`. In scenario 3 a leakage appears between `Turbine 1` and `Condenser 1`. In scenario 4 the nuclear operator closes accidentally `Valve 2` and it gets stuck. In scenario 5 a leakage appears in the cooling system 1. In scenario 6 a strong disturbance

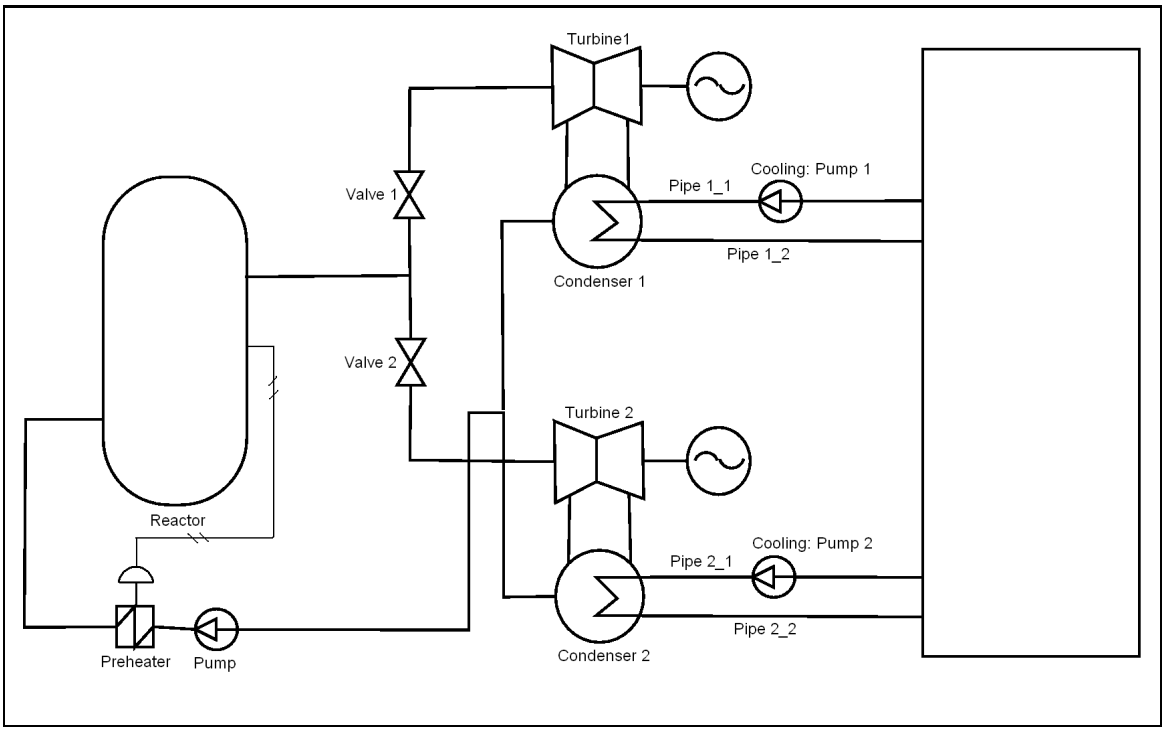

Figure 5.18: PI-diagram of the nuclear power plant Simulink model.

takes place in `Reactor` input power as a result of an earthquake. Every scenario is represented with a matrix with 251 rows and 11 columns where columns are time series of process variables.

Scenario 4 is described in more detail. Figures 5.6 and 5.7 show DERSI operation in this scenario. More detailed time series plots of scenario 4 are shown in Figure 5.19. The most visible deviations are drops in *Turbine 2 Pressure* and *Turbine 2 Temperature* after the `Valve 2` has been closed. The DERSI GUI plots shows the same deviations. In the GUI it can be seen that the trajectory moves from *Normal State* cluster to *Fault C State* cluster which is the state of `Valve 2` closing. As seen in *frame 1*, the rules identify `Valve 2` as closed and recommend either opening of `Valve 2` or opening of the bypass valve. In Figure 5.7 it can be seen that the quantization error of sensor 4 (*s4*) gets small which is another sign of a transition to the fault state of scenario 4.

The simulation does have some deficiencies. For example an unplanned initial transient is seen in early values of *Reactor Upper Pressure* variable. Scenario 6 fault data was too similar to scenario 2 fault so a sensor was not created of this scenario.

5.8.2 DERSI TVO Unit

The first real test of the prototype became possible as data from nuclear power plant simulator was acquired. DERSI GUI of DERSI TVO unit is shown in Figures 5.8 and 5.9. Four scenarios were simulated and in each scenario the simulated fault was slowly developing. All of the simulations have a sampling frequency of 1Hz and a

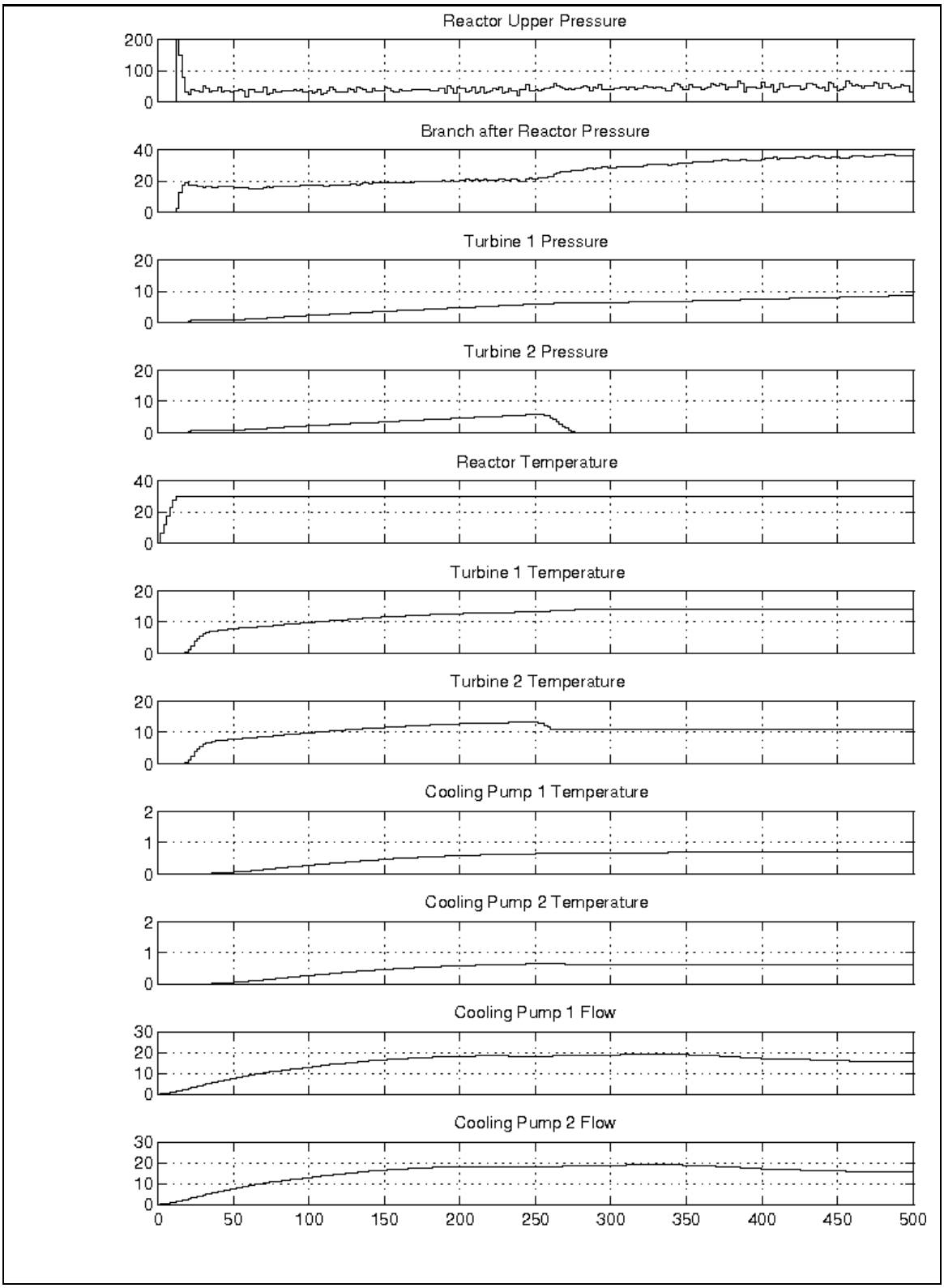

Figure 5.19: Time series plots of Simulink model scenario 4.

fault starts to develop at the instant $t = 1min$ (sample 61). In scenarios 1 and 3 (*d1* and *d3*) the process starts normally and after two minutes a leakage in *reactor coolant pump* begins to develop slowly. In both scenarios the fault becomes serious and the simulation ends into a reactor scram. Difference between these two scenarios is the development speed of the fault. In scenario 2 (*d2*) a *pressure controller* gets broken. This scenario also ends into a serious fault and a reactor scram. Scenario 4 (*d4*) has a series of faults and repairing actions but this scenario is not used in DERSI unit construction because of difficulties in separating data of the different faults.

Three sensors were created in this DERSI unit. Samples 1-60 of *d1* were used for teaching normal state sensor. Samples 61-700 of *d1* and *d2* were used for teaching fault 1 and fault 2 sensors.

5.9 DERSI Version History

DERSI development has lasted for a long time and many versions of the prototype with varying features has been developed. In the beginning it was planned that the prototype will be implemented with the C programming language. After some planning the Matlab programming language became the choice because it would be fast and easy to build the prototype on SOM Toolbox of Matlab.

The first versions of DERSI (*V1.0* and *V1.1*) used very simple data for teaching and testing. The data was generated with sine waves. In *V1.2* the *dersiex* unit building interface was added and DERSI became a platform capable of building a DSS for any application. At the same time a Simulink model was created for providing better test data. Rule numbers and rule priorities were added to the recommendation frame.

In *V1.3* variable choosing lists were added that enabled using datasets with larger amount of variables than 12. Faulty and unnecessary Fourier transform and difference visualizations were removed. Capabilities for generating delayed variable copies were added. Scenario animation capability was added. In *V1.4* some small improvements were made. In version *V1.5* sensor quantization error and progress value visualizations were added.

In *V2.0* the prototype was totally recoded for more reliable and bug-free operation. In version *V2.1* fuzzy inference was implemented. In *V2.2* important normalization bug was fixed that enabled the usage of TVO simulator dataset. Correspondingly a DERSI unit for TVO simulator dataset was built on this version.

Chapter 6

Results and Evaluation

The research problem was formulated as follows:

- **How can self-organizing map (SOM) be utilized in nuclear power plant operator decision support?**

SOM quantization error can be used as an input for both traditional rule-based reasoning and fuzzy rule-based reasoning. SOM U-matrix can be used for process state visualization. Quantization error of this SOM can be used in detection of state transitions and unknown faults. SOM component plane visualizations can be used for correlation based fault detection. Vertical bars can be used for visualization of distance between a SOM input data vector and its BMU. DERSI platform can be used for demonstrating how a nuclear power plant operator could utilize SOM for decision support. Additional approaches for using SOM in decision support were found from literature. They are described in Section 6.5. Utilization of DERSI as a business decision support system (DSS) was also considered and associated discussion is in Section 6.6. DERSI platform needs more testing for showing more of its potential.

6.1 DERSI Platform

The DERSI prototype platform is practically a collection of independent visualization tools and methodologies that are gathered into one user interface (UI) and around one simulation engine. In time critical decision support the tools need to be used fast and efficiently. It was also considered to have separate windows and UIs for each tool. But when a fast analysis of a serious fault needs to be performed, the operator does not have time for unnecessary tasks like organizing windows on the monitor into a convenient arrangement.

As there was much more knowledge about methodologies than about a process to be supported, the prototype developed naturally into a general platform that is a collection of tools and process independent. Processes in different applications have very different requirements for methods and visualizations. Structure and graphical user interface (GUI) of DERSI acts as a constraint for what kind of process data can be used for building a DERSI unit. The platform is usable only for processes that have enough similarities with nuclear power plant process monitoring.

The project of DERSI development begun many years before it was turned into a focus of a Master's thesis study. The project was planned to last from several months to a year. The project continued for longer time than the initially planned duration. After the first versions of the prototype it became apparent that the prototype development should be treated as a continuous process where experimentation is performed and requirements change constantly. Agile programming process should be used in such development.

Later it also became apparent that Matlab is not the best programming language for the prototype because it does not have pointers like C++ or it is not able to pass values by reference like Java. This introduced many problems and forced the author into difficult coding practices. The GUI tools of Matlab and SOM Toolbox were also too slow for the prototype.

6.2 DERSI Unit Building

For a successful DSS development the process to be supported needs to fill the following requirements. *First* the process data has to be as realistic as possible. So the process simulation has to be accurate or the data has to be recorded from a real process. The most trustworthy testing would be possible with real process data and recorded process faults. Obviously this is rarely possible with safety critical processes. *Second* there has to be plenty of process data available. *Third* there has to be a lot of process knowledge available. The best way to incorporate process knowledge in DSS development is to have an application expert participating in the development of the DSS. In this study a nuclear power plant operator would be a suitable application expert.

For to be able to build and test a DSS, second and third requirements need to be filled. For to be able to prove that the DSS methods really provide added value for an operator, all three requirements need to be filled. Process of the Simulink model fills second and third requirements but not the first. A demonstrative prototype unit was created with the Simulink model.

Teollisuuden Voima (TVO) simulator process fills only the first requirement. Author has only limited knowledge of TVO nuclear power plant process and such knowledge was hard to acquire. As the TVO dataset had only three usable process states and not enough details of the underlying process were available, it did not make sense to create a large rule base for the TVO unit. But even with this small rule base it was possible demonstrate that SOM quantization error and process state U-matrix visualization can be used in nuclear power plant operator decision support.

6.3 DERSI Performance

DERSI operation speed is satisfactory with 10-12 variables, but with already 50 variables it becomes slow to operate. In DERSI unit of 50 variables one simulation step can last 5–15 seconds. The simulation snapshot feature can be used in such situation. The prototype performance has not yet been optimized and Matlab limitations slow down the operation. GUI drawing is the slow bottleneck of DERSI.

6.4 DERSI Parts

6.4.1 Diagnoses, Recommendations and Sensors

The rule bases of both DERSI units are so small that they do not show the full potential of fuzzy rule-based reasoning. But the units do demonstrate how it is possible to calculate fuzzy diagnoses and recommendations with fuzzy reasoning from SOM quantization errors and process variable values.

Diagnoses and recommendations of *DERSI Simulink unit* are plausible most of the time. But in a few time periods of some scenarios, inappropriate recommendations or false diagnoses are shown. There are three possible causes for this: 1. SOM sensor produces low quantization error for data of a wrong state, 2. SOM sensor produces high quantization error for data of the right state, 3. rules of the unit are inaccurate or faulty. If some process states cannot be separated into clearly distinguished U-matrix clusters it is likely that faulty reasoning happens in state transitions between these states.

DERSI TVO unit does not have recommendations and its diagnosing performance is evaluated as follows. *Reactor coolant pump leakage* is identified easily with the SOM sensor quantization error, especially near the end of the scenario. *Breakage of pressure controller* and *normal state* data look similar and the states are difficult to separate and identify. Variables 1 and 2 (*P.K.Virta 36* and *P1 Kierrosl.*) correlate in *normal state*, but the correlation vanishes when the *reactor coolant pump leakage* gets stronger. By using SOM sensor quantization errors it is possible to identify *Reactor coolant pump leakage* very early, much earlier than with the process state U-matrix trajectory.

6.4.2 U-Matrix

Previous works have proved the ability of the SOM U-matrix to visualize industrial process state transitions when the SOM is taught with a single time series that contains many process states [36]. As the DERSI process state U-matrix SOM is taught with a combined collection of different process state time series parts, there are many factors and parameters that can increase or decrease the quality and usefulness of the U-matrix visualization. The most important factor is that which teaching samples are chosen from each state.

In *DERSI Simulink unit*, all of the five used process states are separated into clear clusters in the U-matrix visualization as shown in *frame 2* of Figure 5.6. The U-matrix visualization of *DERSI TVO unit* is shown in *frame 2* of Figure 5.8. There is a big smooth area in the upper end of the U-matrix visualization. This area contains clusters but they have unclear borders. In the lower end of the map there are more clear clusters.

This kind of cluster pattern implies that in the U-matrix SOM teaching data there are likely small clusters of data that are euclidically very far away from the rest of the data distribution. Samples of those small clusters act as outliers that stretch the map, cause many map neurons to be situated between teaching data clusters and cause non-optimal learning of the teaching data distribution. The small clusters are

reactor scram states of the process. If samples of the clusters are left outside from U-matrix teaching data, it is likely that the rest of the process states are mapped into more clear clusters. Both the visualizations with *reactor scram states* and the visualizations without those states have their strengths and it might be appropriate to have both kind of visualizations available for the operator.

U-Matrix Teaching Samples

Current version of DERSI platform uses all samples of each SOM sensor teaching data matrix for the process state U-matrix SOM teaching. It is not obvious that such approach produces always a U-matrix visualization where different process states are clearly in different clusters of the U-matrix SOM.

It is an interesting problem that which samples and how many samples of SOM sensor teaching data should be chosen for process state U-matrix SOM teaching to make the U-matrix as informative as possible. For example if 10 samples is taken from one fault for teaching and 100 samples from another fault, the U-matrix visualization will be different than if the sample amounts are exchanged with each other. This problem has not been examined.

One approach for the problem is to choose equal amount of samples from teaching data of each SOM sensor. This approach was used with *Simulink dataset*. But if the faults are slowly developing, there are no clear process state transitions or a process state has only a few samples, then it is not clear that this approach is good. No systematic way was found for choosing teaching data samples from slowly developing faults of *TVO simulator dataset* so experimentation and intuition was used when building the TVO unit.

U-Matrix Evaluation

U-matrix visualization does not provide such information that the operator does not get from SOM sensor quantization errors. The main advantage of the U-matrix visualization is that it is cognitively much easier to follow state transitions from one two-dimensional map visualization than from many plot visualizations. Human error in interpreting the process state from a U-matrix map seems much less likely than when interpreting the process state from large amount of plots. If process states need to be organized two-dimensionally into a monitor, it is convenient to let a SOM decide the organization.

6.4.3 Bar Visualizations

The motivation with bar visualizations was to create a cognitively meaningful state visualization that is easy for the operator to remember. A combination of vertical bars is a clearly distinguishable visual pattern. An average human can remember 5-7 separate things at the same time [23] and this factor limits the usefulness of the bar visualizations. Bar visualizations did not provide much of added value in the built DERSI units.

6.5 Comparison of DERSI with SOM Literature

A large amount of SOM applications can be interpreted as decision support applications. Study of this thesis preceded with other similar studies in the same research facility [4, 6, 5]. It was difficult to find other studies where SOM is utilized in a similar application or in similar ways. Studies with SOM in decision support that were found from literature are listed below. They are from many different applications and some of them are pure method development without a particular application.

One study with a *safety critical DSS* was found. It is a medical diagnostic decision support system for breast cancer detection. Unlike with DERSI, SOM is utilized already in the system design phase where the ordering property of SOM is used in system model selection [27]. One study was found where SOM is used for *regression* in system identification [52]. The study is not application-oriented but applicability of SOM for regression is analyzed thoroughly with three different tasks. One study describes a *recommender system* that uses a clustering algorithm developed from SOM [53]. It is an Internet recommender and unlike DERSI it utilizes system user profile information and constantly changes recommendation logic.

Another study has a *DSS of help desk personnel* that is designed for browsing a database of product problem symptoms and repairing instructions. In that system the symptom descriptions have been transferred to vectors of word frequencies that enable distance calculation between different descriptions. Problem and solution descriptions are associated with these symptom descriptions. They are taught for a SOM that sets similar problems near each other on the map. When searching for prospective solutions for an unknown problem, the description of its symptoms is similarly transferred to a vector of word frequencies. The neighborhood of BMU of this vector can then be searched for likely effective solutions. Because the symptom database is very large, SOM is a suitable method for creating a map of the symptoms. DERSI U-matrix SOM is similar in operation but has different kind of data and is much smaller.

It was found that in addition to the approach of DERSI there are many other ways to combine *fuzzy logic with SOM in decision support*. One study does data clustering with SOM and uses the output of the clustering to generate fuzzy membership functions [54]. In one study a fuzzy rule base is clustered with SOM so that interaction between the rules are reduced in the fuzzy inference [42]. In one study SOM is used for data selection so that one sample is chosen from every single SOM cluster and after that the samples are used for fuzzy rule generation [55].

In two studies SOM is used for map visualization because of its ordering property that is one of its strengths [56, 57]. In the first study *music* can be searched from a SOM so that similar music pieces are neighbors. In the second study *expertise* inside the organization can be searched from a SOM, where one expert is beside another expert with similar knowledge. In these applications the following process is followed: *find X in SOM and explore neighborhood of X for interesting findings*. In DERSI U-matrix SOM neighbors are not interesting, except when the SOM trajectory is about to cross a cluster border.

Two studies were about *customer failures*. Customers unable to pay back credit and customers that are about to change a service provider were identified by their position in SOM [58, 44]. This is similar to fault identification in the current DERSI

applications. In two studies SOM was utilized for vector quantization. In one of them a customer desirability quantity was calculated, and in the other one the coordinates of robot support system movement positions were calculated [59, 43]. Vector quantization property of SOM is not utilized in DERSI.

6.6 Other Applications for DERSI

An additional problem of the study was that in what other applications than nuclear power plant decision support the DERSI platform and its methods could be used. The most significant limitation of DERSI is that the application process data needs to be in form of matrices whose columns are time series variables. This restricts the amount of suitable applications. Data of other applications than TVO nuclear power plant and the Simulink model has not been tested with DERSI. DERSI units could be easily built for applications in other process industries, for example paper industry.

One suitable application for DERSI could be a sort of business DSS whose input variables are associated to sales of various products. The following are examples of diagnosed "fault" states in such application: competing product launch, competing new technology, shortage of liquid finance, warehouse getting filled with product X, warehouse getting empty of product X, growing of product order-to-delivery time, or unfavorable fashion trend. A lot of DSS have been implemented in business context [60].

6.7 Analysis of the Study

As the prototype development in the study has been experimentation where background of the developer is likely to affect into the type of experiments, the study probably produces different results with another developer. Less material than expected were found with the literature study because articles were searched mainly with the *decision support system* term. Later it was realized that decision support applications are often studied without using that term at all.

In solving the main research problem reliability was increased by using three different information sources: prototype development, earlier studies in same laboratory and literature study. DERSI applications were searched only by the author. More applications might have been identified by presenting DERSI more actively to third party researchers from different backgrounds.

6.8 Opportunities for Future Research

6.8.1 Improvements of DERSI

The next step in DERSI development is the installation of DERSI Simulink unit and TVO unit to the premises of TVO nuclear power plant. A TVO nuclear power plant process operator or some other TVO expert will test the units and give feedback from them. This feedback can then be utilized in further studies.

When thinking about the map stretching problem mentioned in Section 6.4.2 it becomes apparent that a new kind of user interface with multiple SOM U-matrices might be useful. In such UI many U-matrix visualizations would be shown in the same screen. Different subsets of existing fault state matrices would be used for teaching different U-matrix SOMs. It is interesting that how much more value a such combination of visualizations would provide for the process operator.

Another new user interface type would be one where the operator has one process state U-matrix of all known states. The operator can zoom to the states. He chooses a rectangular area from the U-matrix and a new U-matrix is formed from teaching data whose BMUs are located in this rectangle. Alternatively there could exist a previously built hierarchy of U-matrix SOMs where each node represents one zoom view.

A third new user interface type would be a variant of U-matrix SOM that is built from scratch in every simulation instant in such way that a neuron in the middle of the map is forced to be a BMU of the current instant input data vector.

One task of an operator of a safety critical process is to question the reliability of all data sources. When using DERSI platform such operator might wonder how did DERSI end up to the diagnoses and recommendations that it shows on the screen. In such situations it might be of value to be able to visualize the inference process and rule truth values so that the operator can confirm that false reasoning has not happened.

6.8.2 New DERSI Units and Studies

If more knowledge and data were provided from TVO nuclear power plant process, even more detailed and plausible DERSI unit could be created. Maybe it would be possible to use data both from TVO simulator and real process faults for building a single DERSI unit.

An interesting environment in finding decision support applicable data is World Wide Web (WWW) 2.0 services. Another similar interesting source is navigation patterns of Web users. This is of special interest of the author.

SOM component planes can be used for monitoring correlations. If component planes are interpreted as vectors and then taught for another SOM, it is possible to group the planes in such way that planes of correlating variables are near each other [61]. It would be interesting to find out, how this increases the operator's ability to detect correlations between variables.

In this study decisions are considered to be individual entities. In many situations sequences of decisions are required and these decisions are related with each other. It would be interesting to study how requirements of a DSS differ and how can SOM be utilized in such situation.

Chapter 7

Summary

Industry and business are full of complicated decision making processes. Probability of human error is high in such decision making. Quality of decisions can be increased and probability of errors can be reduced by providing computerized decision support for the decision maker. The Self-Organizing Map (SOM) is a useful way to visualize high-dimensional and large data sets. The aim of this work is to find approaches for using SOM in nuclear power plant operator decision support and to analyze whether the approaches can be used in decision support of other applications.

Systems and processes are the backbone of an organization or industrial plant. If the systems interact with people of the organization, the connection point is called man-machine interface (MMI). MMI is utilized in making decisions. Decision making processes are especially important as they have large responsibility of success or failure of other processes. Quality control process is important in safety critical processes where the cost of failure is very high. If a system provides support for human decision making, it is a decision support system (DSS). Very often such systems are referred with other names like business intelligence (BI) system.

Software engineering and data mining principles have been followed when implementing a DSS platform prototype, that utilizes SOM and fuzzy rule-based reasoning. The prototype is called DERSI and it is programmed with Matlab programming language and it uses the SOM Toolbox add-on. Used SOM properties are quantization error, U-matrix, component planes and trajectory. Quantization errors of SOM based sensors are used as input for fuzzy inference. Process states are mapped to different areas of the U-matrix visualization and the trajectory head position identifies the current process state. The prototype has a graphical user interface (GUI) that shows visualizations for the prototype user that needs decision support. Results of fuzzy inference are shown as diagnosis and recommendation texts.

Two units of a DSS prototype have been built on this platform. One (Simulink unit) uses data of a Matlab Simulink simulation created by the author before this study. Another one (TVO unit) uses data of the Teollisuuden Voima (TVO) nuclear power plant simulator. These prototypes units demonstrate the possibilities of the methods of the platform. The most important result is that SOM quantization errors can be used in plot visualizations and as an input for fuzzy inference. In TVO unit it is possible to detect a slowly developing fault very early with SOM quantization error, much earlier than with the U-matrix visualization.

Approaches of using SOM in other ways in decision support were searched and found from literature. These were compared with the methods of the prototype. The possibility of using the prototype platform in other applications was analyzed.

7.1 Conclusions

There are many approaches for utilizing SOM in decision support. The decision support applications are numerous and it depends on the application which approaches are most suitable. It is still unclear which of the approaches are most suitable for nuclear power plant operator decision support. When there are many similar methods available it is difficult to choose which one to use. In such case simplicity and reputation of the method and experience of its usage are decisive. The prototype should be tested thoroughly to be sure that it is reliable and to be able to analyze it in more detail. Testing of the prototype with data from another application would make the results of the study more interesting.

References

[1] M. Sirola, *Computerized Decision Support Systems in Failure and Maintenance Management of Safety Critical Processes*, PhD thesis, Helsinki University of Technology, Espoo, Finland, 1999.

[2] E. Alhoniemi, *Monitoring of Complex Processes Using the Self-Organizing Map*, Master's thesis, Helsinki University of Technology, Espoo, Finland, 1995.

[3] O. Simula and J. Kangas, *Neural Networks for Chemical Engineers*, volume 6 of *Computer-Aided Chemical Engineering*, chapter 14: Process Monitoring and Visualization Using Self-Organizing Maps, Elsevier, Amsterdam, Netherlands, 1995.

[4] O. Simula, E. Alhoniemi, J. Hollmén, and J. Vesanto, *Monitoring and Modeling of Complex Processes Using Hierarchical Self-Organizing Maps*, in Proceedings of the 1996 IEEE International Symposium on Circuits and Systems (ISCS), volume Supplement, pages 73–76, 1996.

[5] O. Simula, E. Alhoniemi, J. Hollmén, and J. Vesanto, *Analysis of Complex Systems Using the Self-Organizing Map*, in Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems (ICONIP), pages 1313–1317, 1997.

[6] E. Alhoniemi, J. Hollmén, O. Simula, and J. Vesanto, *Process Monitoring and Modeling Using the Self-Organizing Map*, Integrated Computer-Aided Engineering, 6(1):3–14, 1999.

[7] J. Ahola, E. Alhoniemi, and O. Simula, *Monitoring Industrial Processes Using the Self-Organizing Map*, in Proceedings of the 1999 IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications (SMCia/99), pages 22–27, 1999.

[8] J. Vesanto, *Data Exploration Process Based on the Self-Organizing Map*, PhD thesis, Helsinki University of Technology, Espoo, Finland, 2002.

[9] E. Turban and J. Aronson, *Decision Support Systems and Intelligent Systems*, Prentice Hall PTR Upper Saddle River, New Jersey, USA, 1997.

[10] B. Pershagen, *Light Water Reactor Safety*, Pergamon Books Inc., Elmsford, New York, USA, 1989.

[11] I. Sommerville, *Software Engineering. 6th Edition*, Addison Wesley, Harlow, UK, 2001.

[12] T. Kohonen, *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*, Springer, Berlin, Heidelberg, Germany, 1995.

[13] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, *WEBSOM–Self-Organizing Maps of Document Collections*, Neurocomputing, 21(1-3):101–117, Elsevier, Amsterdam, Netherlands, 1998.

[14] R. Banerji, *Artificial Intelligence, A Theoretical Approach*, Elsevier, Amsterdam, Netherlands, 1985.

[15] L. Zadeh, *Fuzzy Sets*, Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers. World Scientific Pub Co Inc, 1996.

[16] S. Kartalopoulos, *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications*, Wiley-IEEE Press, 1997.

[17] M. Conforti, *Decision Support Systems for Medical Diagnosis*, Information Technology Applications in Biomedicine, pages 25–26, 1999.

[18] P. Lillrank, *Quality Management Basic Course TU-22.302*, Teaching Material, Helsinki University of Technology, Espoo, Finland, 2004.

[19] E. Kreyszig, *Advanced Engineering Mathematics, 8th Edition*, Wiley, New York, USA, 1999.

[20] R. Stevens, *Systems Engineering: Coping with Complexity*, Prentice Hall Europe, 1998.

[21] W. Stevenson and M. Hojati, *Operations Management*, McGraw-Hill/Irwin, Boston, USA, 2005.

[22] A. Jashapara, *Knowledge Management: An Integral Approach*, Pearson Education, 2004.

[23] A. Huczynski and D. Buchanan, *Organizational Behaviour: An Introductory Text*, Financial Times/Prentice Hall, 2007.

[24] N. Bahr, *System Safety Engineering and Risk Assessment: a Practical Approach*, CRC, 1997.

[25] R. Guldner and U. Giese, *EPR Becomes Reality at Finland's Olkiluoto 3*, CHIMIA International Journal for Chemistry, 59(12):966–969, Swiss Chemical Society, 2005.

[26] M. Dowd, J. McDonald, and J. Schuh, *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*, Addison-Wesley Professional, 2006.

[27] D. West and V. West, *Model Selection for a Medical Diagnostic Decision Support System: a Breast Cancer Detection Case*, Artificial Intelligence In Medicine, 20(3):183–204, Elsevier, Amsterdam, Netherlands, 2000.

[28] R. Martin, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall PTR Upper Saddle River, New Jersey, USA, 2003.

[29] A. Wilson, *Marketing research: an integrated approach*, Prentice Hall/Financial Times, 2003.

[30] X. Faulkner, *Usability Engineering*, Macmillan, 2000.

[31] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.

[32] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, *CRoss Industry Standard Process for Data Mining Crisp-DM 1.0.*, 1999.

[33] S. Haykin, *Neural Networks, a Comprehensive Foundation*, Prentice Hall, 1999.

[34] The Mathworks, Inc., *Using Matlab Version 5*, 1999.

[35] G. Barreto, A. Araujo, and H. Ritter, *Time in Self-Organizing Maps: An Overview of Models*, International Journal of Computer Research, 10(2):139–179, 2001.

[36] M. Sirola and J. Vesanto, *Utilization of Neural Methods in Knowledge-Based Decision Support Systems - State Monitoring as a Case Example*, in Proceedings of the 2000 International Conference in Modeling, Identification and Control, Innsbruck, Austria, 2000.

[37] K. Kiviluoto, *Predicting Bankruptcies with the Self-Organizing Map*, Neurocomputing, 21(1-3):191–201, Elsevier, 1998.

[38] R. Hakala, T. Simila, M. Sirola, and J. Parviainen, *Process State and Progress Visualization Using Self-Organizing Map*, Lecture Notes in Computer Science, Springer, 2006.

[39] R. Stuart and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey, 1995.

[40] R. F. H. Young, *University Physics, 9th Edition*, Addison-Wesley, 2000.

[41] S. French, J. Bartzis, J. Ehrhardt, J. Lochard, M. Morrey, N. Papamichail, K. Sinkko, and A. Sohier, *RODOS: Decision Support for Nuclear Emergencies*, Recent Developments and Applications in Decision Making, pages 379–394, 2000.

[42] P. Chang, C. Liu, and Y. Wang, *A Hybrid Model by Clustering and Evolving Fuzzy Rules for Sales Decision Supports in Printed Circuit Board Industry*, Decision Support Systems, 42(3):1254–1269, Elsevier, Amsterdam, Netherlands, 2006.

[43] Y. Hayakawa, T. Ogata, and S. Sugano, *Flexible assembly work cooperating system based on work state identifications by a self-organizing map*, IEEE/ASME Transactions on Mechatronics, volume 9, 2004.

[44] J. Huysmans, B. Baesens, J. Vanthienen, and T. Gestel, *Failure Prediction with Self Organizing Maps*, Expert Systems with Applications, 30(3):479–487, 2006.

[45] N. Kasabov, L. Erzegoveri, et al., *Hybrid Intelligent Decision Support Systems and Applications for Risk Analysis and Prediction of Evolving Economic Clusters in Europe*, Future Directions for Intelligent Systems and Information Sciences, Physica Verlag (Springer Verlag), 2000.

[46] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, *SOM Toolbox for Matlab 5*, Report A57, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 2000.

[47] M. Sirola, G. Lampi, and J. Parviainen, *Neuro Computing in Knowledge-Based Decision Support Systems*, in Proceedings of the 2004 EHPG-Meeting of OECD Halden Reactor Project, Sandefjord,Norway, 2004.

[48] M. Sirola, G. Lampi, and J. Parviainen, *Using Self-Organizing Map in a Computerized Decision Support System*, in Proceedings of the 2004 International Conference on Neural Information Processing (ICONIP), Calcutta, India, 2004.

[49] M. Sirola, G. Lampi, and J. Parviainen, *SOM Based Decision Support in Failure Management*, International Scientific Journal of Computing, 3(4):124–130, 2005.

[50] M. Sirola, G. Lampi, and J. Parviainen, *Failure Detection and Separation in SOM Based Decision Support*, in Proceedings of the 2007 Workshop on Self-Organizing Maps (WSOM), Bielefeld, Germany, 2007.

[51] G. Lampi, *Building a Process Model to Produce Data for The Computerized Decision Support System Using Self-Organizing Map*, Special Study Report, Helsinki University of Technology, Espoo, Finland, 2004.

[52] G. Barreto and A. Araujo, *Identification and Control of Dynamical Systems Using the Self-Organizing Map*, IEEE Transactions on Neural Networks, volume 15, 2004.

[53] W. Lihua, L. Lu, L. Jing, and L. Zongyong, *Modeling User Multiple Interests by an Improved GCS Approach*, Expert Systems With Applications, 29(4):757–767, Elsevier, Amsterdam, Netherlands, 2005.

[54] C. Yang and N. Bose, *Generating Fuzzy Membership Function With Self-Organizing Feature Map*, Pattern Recognition Letters, volume 27, 2005.

[55] I. Nakaoka, K. Tani, Y. Hoshino, and K. Kamei, *A Portfolio Selection by SOM and an Asset Allocation of Risk/Nonrisk Assets by Fuzzy Reasoning Using the Selected Brands*, in Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, volume 3, 2005.

[56] Z. Huang, H. Chen, F. Guo, J. Xu, S. Wu, and W. Chen, *Expertise Visualization: An Implementation and Study Based on Cognitive Fit Theory*, Decision Support Systems, 42(3):1539–1557, Elsevier, Amsterdam, Netherlands, 2006.

[57] E. Pampalk, S. Dixon, and G. Widmer, *Exploring Music Collections by Browsing Different Views*, Computer Music Journal, 28(2):49–62, MIT Press, 2004.

[58] Y. Kim, H. Song, and S. Kim, *Strategies for Preventing Defection Based on the Mean Time to Defection and Their Implementations on a Self-Organizing Map*, Expert Systems, 22(5):265, 2005.

[59] W. Yan, C. Chen, and L. Khoo, *A Web-Enabled Product Definition and Customization System for Product Conceptualization*, Expert Systems, 22(5):241–253, Blackwell Synergy, 2005.

[60] P. Kotler, *Marketing Management, 11th edition*, Pearson Education, 2002.

[61] J. Vesanto and J. Ahola, *Hunting for Correlations in Data Using the Self-Organizing Map*, in Proceedings of the 1999 International ICSC Congress on Computational Intelligence Methods and Applications (CIMA), 1999.

Appendix A

Rules of DERSI Simulink Unit

Rules of DERSI Simulink unit are listed below. Consequents of *diagnosis rules* are logical statements and consequents of *recommendation rules* are imperatives. Consequents of *diagnosis rules* 2, 3, 4 and 5 are antecedents of *recommendation rules* 7, 8, 9 and 10.

```
*****************************************
DIAGNOSIS RULES:

DIAGNOSIS RULE 1:
IF   -Inf < -Inf <    Sensor 1 qerror    < 0.25 < 2.5

THEN   Normal State, Sig: 85


DIAGNOSIS RULE 2:
IF   -Inf < -Inf <    Sensor 2 qerror    < 0.25 < 2.5

THEN   Leak between Reactor and Preheater, Sig: 85


DIAGNOSIS RULE 3:
IF   -Inf < -Inf <    Sensor 3 qerror    < 0.25 < 2.5

THEN   Leak between turbine 1 and condenser primary 1, Sig: 85


DIAGNOSIS RULE 4:
IF   -Inf < -Inf <    Sensor 4 qerror    < 0.25 < 2.5

THEN   Valve 2 closed, Sig: 85


DIAGNOSIS RULE 5:
IF   -Inf < -Inf <    Sensor 5 qerror    < 0.25 < 2.5
```

```
THEN   Leak in cooling between Pipe1_1 and condenser secondary 1,
          Sig: 85

**************************************
NORMAL RECOMMENDATION RULES:


RECOMMENDATION RULE 1:
IF   2 < 3 <    Variable 1 value    < 10 < 12

THEN   Increase speed of feedwater pump, Sig: 80


RECOMMENDATION RULE 2:
IF   0.8 < 1 <    Variable 1 value    < 3 < 5

THEN   Activate the auxiliary feedwater system, Sig: 90

RECOMMENDATION RULE 3:
IF   -Inf < -Inf <    Variable 1 value    < 1 < 2

THEN   Activate boron injection system, Sig: 88


RECOMMENDATION RULE 4:
IF   25 < 32 <    Variable 2 value    < 33 < 34

THEN   Open bypass valve, Sig: 90


RECOMMENDATION RULE 5:
IF   32 < 33 <    Variable 2 value    < 36 < 37

THEN   Open relief valve and initiate auxiliary feedwater system,
          Sig: 95

RECOMMENDATION RULE 6:
IF   33 < 36 <    Variable 2 value    < Inf < Inf

THEN   Prepare for possible reactor shutdown, Sig: 93

******************************
DIAGNOSIS RECOMMENDATION RULES:

RECOMMENDATION RULE 7:
```

```
IF   0.5 < 0.9 <    Diagnosis 2 truthness    < Inf < Inf

THEN   Decrease speed of feedwater pump, Sig: 80

RECOMMENDATION RULE 8:
IF   0.5 < 0.9 <    Diagnosis 3 truthness    < Inf < Inf

THEN   Close admission valve 1, Sig: 90


RECOMMENDATION RULE 9:
IF   0.5 < 0.9 <    Diagnosis 4 truthness    < Inf < Inf

THEN   Open admission valve 2, Sig: 80

RECOMMENDATION RULE 10:
IF   0.5 < 0.9 <    Diagnosis 5 truthness    < Inf < Inf

THEN   Insert control rods, Sig: 90
```