



Universitat de Girona

**INTEGRATION OF KNOWLEDGE-BASED,
QUALITATIVE AND NUMERIC TOOLS FOR REAL
TIME DYNAMIC SYSTEMS SUPERVISION**

Joaquim MELÉNDEZ i FRIGOLA

**ISBN: 84-8458-104-7
Dipòsit legal: GI-1216-2001**

University of Girona - Universitat de Girona



Department of Electronics, Computer Science and Automatic Control
Departament d'Electrònica Informàtica i Automàtica (E.I.A.)

Integration of Knowledge-Based, Qualitative, and Numeric Tools for Real Time Dynamic Systems Supervision

by

Joaquim Melendez Frigola

Supervisors :

Dr. Josep Lluís de la Rosa

Dr. Josep Aguilar-Martin.

Girona

December, 1997

**To my wife Maria,
my parents Joaquim and Roser
and my sisters Cristina and Rosa.**

This work is submitted as the doctoral thesis for the degree of Doctor of Philosophy at the Department of Electronics, Computer Science and Automatic Control (“Doctor Enginyer Industrial al Departament d’Electrònica Informàtica i Automàtica”) of the University of Girona.

The thesis describes original work in the area of Computer Aided Supervisory Systems Design carried out at the University of Girona and partially at the Laboratoire d’Architecture et d’Analyse de Systèmes (LAAS-CNRS) in Toulouse (France).

Acknowledgements

At the end of this work, the author realises that despite of the individual effort put into it, many persons contributed to improve the quality of results and the final work. This text intends to reflect the time and advice of all those people who the author is indebted to. These persons, include especially, the directors and supervisors of the work, Josep Lluís de la Rosa, lecturer at the University of Girona, and Josep Aguilar Martin, professor at the University of Girona and directeur de recherche au LAAS-CNRS. They have guided this work with their support, contributions and experience. I am also indebted to Antoni Ligeza from the Institute of Mining and Metallurgy in Kraków (Poland) for his formal correction and encouragement during the time he spent in Girona.

I do not forget the months at the Laboratoire d'Architecture et d'Analyse de Systèmes (LAAS-CNRS) in Toulouse (France) and the encouragement received from B. Dahhou, who also has been a reviewer of this work, J.B. Pourciel and the valuable support of Gilles Roux, Bernard Franc and Joseph Urraca.

Furthermore, I want to thank the persons who accepted to review the contents of this text and that have contributed to improve the final work with their valuable contributions and advice. This is the case of Dr. K Bousson (from the Universidade da Beira Interior in Portugal), Dr. M. Szymkat (from the Insitute of Automatics, at the University of Mining and Metallurgy in Kraków, Poland), and Dr. J. Puyol (from the Institut d'Investigació en Intel·ligència Artificial del Consejo Superior de Investigaciones Científicas in Spain, IIIA-CSIC).

My sincere gratitude to the unconditional support of all members of my group, eXiT (enginyeria de Control i Sistemes iTel·ligents) at the University of Girona, for the amount of time, effort and patience kindly inverted in this work. Particular thanks to the colleagues Xavier Llauro, Joan Colomer who tested some parts of the work as a useful tool for their research purposes. I also want to mention the students who contributed with their works to the elaboration of some of the tools and examples described in this work. This is the case of Alfons Sabat, David Gutierrez and Toni Martinez.

I am also indebted to Patricia, who accurately chose the last words in this text.

This brief page is to thank all of them.

Contents

Acknowledgements	iii
Contents	iv
Preface	vii
1. INTRODUCTION AND OVERVIEW	1
1.1. General Introduction.	1
1.2. Expert Supervision. General Scope.	3
1.3. Introduction to the thesis.	4
1.3.1. General Problem Description.	4
1.3.2. Outline of the proposed approach.	6
1.4. Objective and thesis of the work.	7
1.5. Conclusions.	8
2. EXPERT SUPERVISION	10
2.1. Introduction.	10
2.2. Supervision, fault detection and diagnosis : Terminology and definitions.	11
2.3. Fault detection: structures and methodologies.	13
2.3.1. Model-based fault detection.	15
2.3.2. Signal-based fault detection.	16
2.3.3. Knowledge-based fault detection.	17
2.4. Knowledge-based fault diagnosis.	19
2.4.1. Knowledge-based methods used in fault diagnosis.	20
2.4.2. Diagnosis strategies.	21
2.5. Implementation of expert supervisory systems.	22
2.5.1. Assisting expert supervision design.	23
2.6. Conclusions.	25
3. METHODS FOR EXPERT SUPERVISORY SYSTEMS	26
3.1. Introduction: Necessity of AI methods for process supervision.	26
3.1.1. Heterogeneous, imprecise and uncertain data.	27
3.1.2. Process behaviour and expert knowledge.	28
3.2. Knowledge representation techniques.	28
3.2.1. Logical formalisms.	29
3.2.2. Rule-based systems.	29
3.2.3. Graphs and including semantic networks.	31
3.2.4. Frames and object oriented representations.	32
3.3. Qualitative reasoning.	33
3.3.1. Qualitative modelling and simulation.	34
3.4. Artificial Intelligence in Control Systems.	36

3.5. Artificial Intelligence in Process Monitoring and Supervision.	37
3.5.1. Integration problem.	38
3.5.2. Imprecision in temporal references.	39
3.5.3. Benefits of using OOP in supervisory tasks.	39
3.6. Conclusions.	40
4. CASSD - COMPUTER AIDED SUPERVISORY SYSTEMS DESIGN FRAMEWORKS	41
4.1. Necessity of CASSD frameworks.	41
4.2. Antecedents.	42
4.2.1. Steps towards a tools-based architecture.	42
4.2.2. The reference model for CACE open environments.	45
4.2.3. CACE and CACSD packages.	47
4.2.4. MATLAB/Simulink as a CACSD framework.	48
4.3. Coupled CACSD frameworks.	50
4.3.1. Knowledge-based systems and MATLAB/Simulink.	52
4.3.2. Characteristics of Expert Systems for CACSD.	53
4.4. Proposal for a Computer Aided Supervisory System Design (CASSD) framework.	55
4.4.1. Requirements for a CASSD framework.	55
4.4.2. Selection of a platform.	58
4.4.3. Other frameworks with similar capabilities.	58
4.5. Conclusions.	59
5. INTEGRATION OF TOOLS FOR SUPPORTING SUPERVISORY SYSTEMS DESIGN	61
5.1. Introduction.	61
5.2. Sharing Information between tools : <i>Object-variables</i>.	62
5.2.1. Using process variables for reasoning.	62
5.2.2. Process variables at different levels of abstraction : <i>Object-variables</i> .	64
5.2.3. About objects in MATLAB-Simulink. Containers.	66
5.2.4. Classes of <i>object-variables</i> .	68
5.2.5. Embedding objects into MATLAB/Simulink. Technical viability.	70
5.3. Abstraction tools.	71
5.3.1. Significant information from process variables.	72
5.3.2. A taxonomy/selection of abstraction tools.	75
5.3.3. Kinds of abstractors.	76
5.3.4. Abstraction tools and <i>object-variables</i> .	80
5.4. ALCMEN, A representation language.	81
5.4.1. Fundamentals of ALCMEN.	81
5.4.2. Static operators	84
5.4.3. Dynamic operators (time depending relationships).	88
5.4.4. Qualitative representation of numerical signals.	88
5.4.5. ALCMEN and qualitative reasoning.	89
5.4.6. Implementation in MATLAB/Simulink.	90
5.5. Expert System CEES into MATLAB/Simulink.	92

5.5.1. CEES features. Advantages and drawbacks.	93
5.5.2. SimCEES : Implementation of an ES in MATLAB/ Simulink.	95
5.5.3. Expert Reasoning with SimCEES.	97
5.5.4. Exportability of rules. LabCEES, the stand-alone application.	99
5.6. The global framework.	100
5.6.1. Architecture and tools.	100
5.6.2. Knowledge representation.	102
5.7. Conclusions.	104
6. SOME ILLUSTRATIVE EXAMPLES IN USING THE CASSD FRAMEWORK	106
6.1. Introduction	106
6.2. Fault diagnosis of a simulated plant.	107
6.2.1. Laboratory plant description.	107
6.2.2. Implementation of a knowledge-based fault detection system in the CASSD framework.	109
6.3. Qualitative estimation of a variable.	114
6.4. Qualitative estimation of temperature in the furnace of a Municipal Solid Waste Incineration plant.	121
6.4.1. Plant description : the furnace.	121
6.4.2. About MSW quality.	123
6.4.3. Qualitative estimation of temperature.	124
6.5. Validation of a knowledge base when designing an ES for fault diagnosis.	127
7. CONCLUSIONS AND FUTURE WORK	131
7.1. Conclusions.	131
7.2. Future Work.	135
8. GLOSSARY	137
9. BIBLIOGRAPHY	139

Preface

This dissertation contains the research work on integrating artificial intelligence technologies to assist process supervisory design tasks developed by the author from 1994 to 1997 at the Departament d'Electrònica Informàtica i Automàtica of the University of Girona. This work has been developed within two research projects supported by the Spanish government ("Desarrollo de Sistemas de Control Inteligentes para procesos Industriales"- CICYT.TAP 93/0596, and "Plataformas Integradas de CAD de Supervisión y metodologías" CICYT.TAP96-1114-C03-03). The work has been partially developed in the Laboratoire d'Architecture et d'Analyse de Systemes (LAAS-CNRS) in Toulouse (France).

The work developed during this period of time has been focused on integrating artificial intelligence techniques to manage information coming from dynamic systems. The work shows how artificial intelligence can be used in process supervision and monitoring for reasoning about process behaviour by means of expert knowledge representation. A framework is developed to assist these tasks avoiding integration problems during the development of supervisory applications. This framework facilitates the use different tools, from the artificial intelligence domain, for reasoning about perception of process behaviour.

The general theme in the thesis is about integration of tools to assist supervisory systems design. The work has been restricted to knowledge-based systems. Interest is centred on knowledge representation and reasoning about dynamic systems for developing supervisory structures. For the implementation of this framework, the shell CEES has been used to represent rule-based expert systems, while qualitative relationship can be implemented by means of a qualitative representation language called ALCMEN. These tools are interfaced with numerical information by means of a set of tools designed to abstract qualitative representation of numerical data, i.e. abstraction tools. Integration is performed by means of an object oriented approach. This thesis is focused on those topics according to the following structure.

- Introduction and Overview. A general description of the problem and proposed solution. Basic topics and terminology is introduced in the context of this work. The goal of this work is also presented.
- Expert Supervision. The increasing interest in this domain and the necessity of taking benefit of expert knowledge to improve industrial process evolution is introduced. Actual methodologies and strategies for diagnosis and supervision are discussed. The complexity in designing supervisory structures leads to the necessity of using specialised tools and frameworks for assisting such designs.
- Methods for expert supervision. A review of technologies, methods and tools used in expert supervisory systems implementation. The majority comes from the AI domain and its usability and applicability in the expert supervision domain is discussed.
- Frameworks for Computer Aided Supervisory Systems Design (CASSD). The necessity of a framework for dealing with expert knowledge representation in supervisory task is discussed. The antecedents and previous works in the control domain are reviewed and analysed to define the characteristics of the proposal.
- Integration of tools for CASSD in MATLAB/Simulink. Object-oriented approach is proposed as integration methodology. Several tools from the AI domain are proposed and implemented to configure a CASSD framework. A qualitative representation language, called ALCMEN, and a shell for dealing with expert systems, called CEES, are selected to deal with knowledge representation and processing. Interface numeric to qualitative is performed by different algorithm called *abstraction tools*.
- Design of Supervisory Structures. Examples of using the framework for designing simple fault detection applications, qualitative modelling and knowledge base validation applications are presented in this chapter.

Partial conclusions are added at the end of each chapter and emphasised in the last chapter, were also further work and open lines are remarked.

1.

Introduction and Overview

1.1. General Introduction.

The increasing use of computers in engineering activities has originated the apparition of numerous packages or frameworks especially conceived to facilitate engineering developments. Some reasons for extensive application of computers in engineering decision and design problems are quite obvious; computer aided tools make the work of engineers both easier and faster. This means that increasingly more complex engineering problems become solvable in reasonable short time, while the effort devoted to solve these problems remains limited mostly to some conceptual work. Some other reasons include the necessity of improving the of quality of engineering solutions, including robustness, reliability and safety and lowering down the cost of manpower spent on developing particular solutions. Moreover, the systems developed with use of computer software can easily be simulated, analysed and reused.

Because of these reasons, the acronym CA (Compute Aided) has been used as a prefix for designing software packages developed to assist engineering tasks. Some principal areas of interest in applying computers for assisting engineers include domains as:

- CAD - Computer Aided Design.

- CADS - Computer Aided Decision Support, also known as DSS - Decision Support System.
- CAE - Computer Aided Engineering.
- CACE - Computer Aided Control Engineering.
- CAM - Computer Aided Manufacturing.
- CASE - Computer Aided Software Engineering.
- CACSD - Computer Aided Control Systems Design,

The application of computers becomes more important in everyday engineering activity. Particularly, control engineering community has been actively pursuing avenues for supporting multi-disciplinary design activities for several years. Consequently, the research activity in this field has been increased to support design, implementation and maintenance of control projects. CACSD packages have been developed following these lines. They incorporate representation, analysis and design facilities. Therefore, these packages are dotted of friendly user interfaces, representation capabilities and a wide number of numerical algorithms (modelling and simulation capabilities) to facilitate control systems design, test and validation.

Nowadays, control engineers activities in the industry are not only reduced to control systems design but their activity must also take into account all process behaviour, especially when dealing with complex systems (non-linear, coupled, time dependent, etc.). In such cases, control and process engineers must work together to ensure global quality, safety and reliability in both, product, and process behaviour. This is process supervising. From the control engineers point of view supervisory task consists in closing a high level control loop. Supervisory systems will be designed to observe the process (process measures, visual observations, indices and so on), to decide about its behaviour according to a template (predefined normal operating conditions) and to perform or suggest specific actions (control reconfiguring, set point change, etc.) when necessary. The complexity in designing supervisory systems makes necessary the use of all available information about instantaneous process behaviour. Process measures given by sensors, estimations obtained from models (numerical or qualitative) and expert knowledge obtained from operators and process engineers, must be merged to perform supervisory strategies in the best way. Thus, Expert Supervision is used in this text to design such supervisory strategies that take benefit of expert knowledge about process behaviour.

Following this line, control engineers working in the domain of supervisory systems are also needed for specific software packages to assist their activities. This thesis puts forward several particular solutions in this relatively new area, i.e. Computer Aided Supervisory Systems Design- CASSD.

1.2. Expert Supervision. General Scope.

Nowadays, automatic control applications in the domain of process supervision are restricted to decision making, including human operator, into the feedback loop [Millot, 1996]. In fact, the functionality of Supervisory Control And Data Acquisition (SCADA) packages, widely used in the industry, are restricted to monitoring and alarm generation tasks. Final decisions about the validity of these alarms and actions to perform in the process are done by expert operators in the plant. Design difficulties of supervisory systems increase with process complexity. Coupled systems, high order dynamics, and non-linearity are common situations in industrial process. In such situations, it is difficult to obtain accurate models for applying traditional model-based techniques for designing supervisory systems and knowledge-based approaches can be applied taking benefit of process operators and process engineers' experiences to identify specific situations. Therefore, Expert Supervision is an active research line oriented to take advantage of expert knowledge of process engineers and plant operators to automatically decide about process behaviour and to propose adequate actions or changes in the set points, controllers parameters or reconfiguring strategies. Artificial intelligence (AI) technologies are applied to deal with this expert knowledge inside computers because of its capabilities for:

- Knowledge representation. This is the interpretation of human knowledge and translation into computers. AI proposes several methods for structuring and organising expert knowledge into knowledge bases (KBs). Rule-based systems, also called expert systems (ES), are the most common example of applications used to represent expert knowledge.
- Reasoning. This is the capability of using the expert knowledge to infer conclusions as a consequence of specific situations.
- Manipulation of heterogeneous data. The use of AI technologies allows to manage qualitative and symbolic information mixed with numerical data and methods.

Those capabilities are extremely useful in expert supervision because of the different origin of information to be used. Expert supervisory applications are designed to reason about process variables, i.e. signals that represent physical variables in the process and provide information about process behaviour. These signals can be measures, numeric data obtained directly from real process or controllers, or estimations, obtained from models or relations. According to the kind of relationship or models used to estimate a process variables we can differentiate between qualitative, or symbolic, and numeric estimations. These signals can be manipulated or processed to isolate significant information, i.e. abstracted information.

The complexity of designing, testing and validating the expert supervisory systems is basically due to the variety of tools and data to be used. Difficulties in integration are the main reason of failure in supervisory strategies implementation. This pitfall will decrease if all the available facilities to be used are available together in a framework, avoiding integration inconvenient, especially oriented to assist expert supervisory systems design.

1.3. Introduction to the thesis.

1.3.1. General Problem Description.

Expert Supervisory Systems design involves manipulation of heterogeneous information such as signals, knowledge, qualitative data and so on. All of them are representations of process behaviour that must be taken into account when designing supervisory systems. Numerical data comes basically from direct measures of process variables or from estimations supplied by numerical models or equations. Usually, numerical data is the kind of information supplied from process periodically, at every sampling time. This kind of information can be saved, processed and represented in several ways to be analysed. On the other hand knowledge and qualitative appreciation of process dynamics come from heuristics or observations of representative situations done by process engineers (Fig. 1.1). In industrial applications the existing knowledge representation procedures for assisting experts in translating and structuring this knowledge into computers are basically reduced to production rules systems. For example, fuzzy based controllers or industrial ES are in this line. Reasoning about dynamic systems involve working with temporal restrictions as expiring data validity and limitations in the response time. Periodically (at each sampling time), ES input data is actualised and deductions about them must be performed.

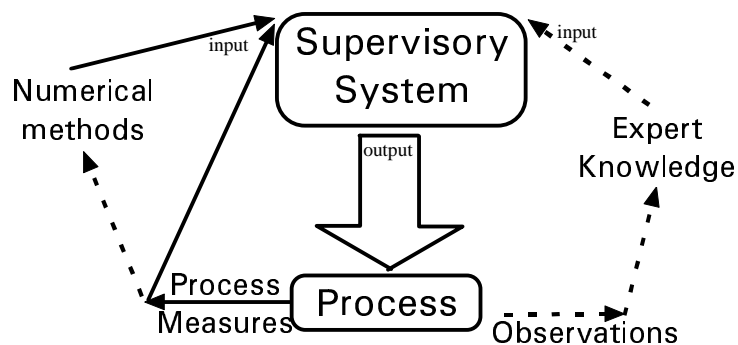


Fig. 1.1 Representation of information supplied to a supervisory system.

Therefore, design of a knowledge-based system to reason about dynamic data involves an important task on data analysis and features extraction to match description of situations given by experts. Main drawbacks are in obtaining perceptual information from process variables, in order to interface expert KBs, defined from human perception of process behaviour, and measures, supplied by instruments and numerical methods. Consequently, the design of expert supervisory systems consists of an iterative procedure until specifications are reached.

The definitive application sometimes involves the election of adequate description of input and parameters tuning by trial and error. This iterative procedure, represented in Fig. 1.2, is necessary, not only in the designing step, but also in previous tasks for defining adequate input and output of expert supervisory system.

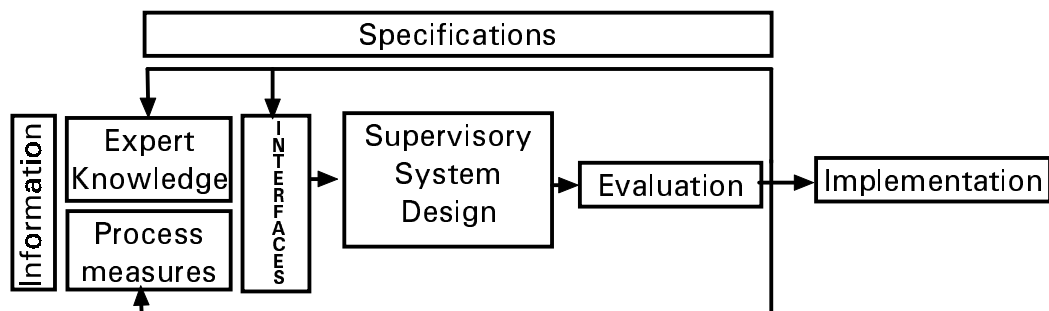


Fig. 1.2 Supervisory systems design is an iterative task.

Input must be defined in number, type and quality. The number of input can be restricted to according to process limitations (location, observability, transmission) or economical restrictions. Then, estimation capabilities (indirect measures, modelling) can be used as alternative. Data typology is always submitted to KB to be used, due to human subjectivity in the observations. These observations of process behaviour are strongly time dependent and, consequently, process variables supplied to inference engines must incorporate this dependency. Thus, the use of abstracted significative time representations of process variables at different levels will be very useful and, therefore, the exact shape of those trends must be defined to match expert interpretations. Certainty and imprecision is another factor to be taken into account in some kind of measures or estimations that are subject to unknown factors (perturbations, unknown parameters dependencies, deviations, time dependency, failures, and so on). Similar specification of output must be performed if they are used to as a high level feedback loop to propose actions to improve global system performances.

It is evident that different tools will be used to deal with this variety of information. As a consequence, interfacing problems between applications and data type mismatch will occur at the same time that users (supervisory systems designers) will operate with distinct user interfaces and applications front-end. This work is centred on providing a set of tools integrated into a framework to facilitate management of information when designing expert supervisory systems.

1.3.2. Outline of the proposed approach.

The proposal presented in this thesis is to provide designers of knowledge based supervisory systems of dynamic systems with a framework to facilitate their tasks avoiding interface problems among tools, data flow and management. The approach is thought to be useful to both control and process engineers in assisting their tasks. The use of AI technologies to diagnose and perform control loops and, of course, assist process supervisory tasks such as fault detection and diagnose, are in the scope of this work. Special effort has been put in integration of tools for assisting expert supervisory systems design. With this aim the experience of Computer Aided Control Systems Design (CACSD) frameworks have been analysed and used to design a Computer Aided Supervisory Systems (CASSD) framework. In this sense, some basic facilities are required to be available in this proposed framework:

- Abstraction Tools . These are tools for signal processing, representation and analysis to obtain significative information.
- To deal with process variables, measures or numerical estimations, and expert observations, with uncertainty and imprecision.
- Expert knowledge representation at different levels by using a rule-based system or simple qualitative relations.
- Modularity and encapsulation of data and knowledge would be useful for structuring information.
- Graphical user interface to manage all those facilities in the same environment as actual CACSD packages.

An existing commercial framework has been chosen, MATLAB/Simulink, to add those facilities in order to assist expert supervisory systems design. MATLAB/Simulink has been selected because of its proximity to control engineers and easy to use graphical user interface. Simulink blocks are used in two ways : to encapsulate information (data and methods) and to force engineers to structure they knowledge in a graphical representation.

Several tools from the AI domain have been added as Simulink ToolBoxes to deal with abstracted information, qualitative relationship and rule-based ES. Simple and intuitive qualitative relationship can be implemented by means of a

block-based qualitative representation language called ALCMEN. An ES shell, called CEES, has also been embedded into MATLAB/Simulink as a block to allow modularisation and partition of large expert KBs. Finally, the numeric to qualitative interfaces is performed by a set of algorithms, called *abstraction tools*, encapsulated also in Simulink blocks. The functionality of the whole framework is able due to the use of object oriented approach in the development and implementation of those tools.

1.4. Objective and thesis of the work.

Current computer aided environments for assisting engineering tasks have reached their majority in the domain of numerical data processing and representation. All widely used systems provide well-developed and user-friendly tools for numerical data acquisition, all types of mathematical calculation and variety of diagrams for output data representation. Moreover, most of the systems provide graphical user interface including symbolic block diagram editor which offers an easy way for programming and turns the complex engineering task to simpler manipulation on icons. This makes the design and analysis process very intuitive and close to physical ‘interpretation’, i.e. realistic systems manipulation.

On the other hand, the current computer aided systems still lack of consistent, intuitive, and adequate support for symbolic and qualitative data and knowledge management. Such kind of knowledge constitutes however a very important component of more complex systems design and analysis tasks such as process monitoring, supervision, diagnosis, safety and reliability analysis and etc. In this thesis an attempt is undertaken to make steps towards integration of tools for expert supervision, including once for qualitative and symbolic data representation and management and symbolic knowledge processing. The main research objectives of this work include the following points :

1. Incorporation of object-variables into classical numerical data processing system. The aim is to allow structural qualitative and symbolic knowledge representation. Complex information is encapsulated in a single source/sink structure, called *object-variable*, providing methods for knowledge access and processing.

2. Implementation of selected particular tools for qualitative and symbolic knowledge representation and interfacing. Higher abstract level information processing based on the introduced *object-variables*.

3. Embedding an object oriented rule-based expert system into a classical CACSD framework in order to provide high level knowledge processing facilities based on the domain of expert knowledge, heuristics, and logic.

The principal thesis of this work is that : *Integration of object oriented knowledge representation and processing into classical numerical methods based computer aided control system design contributes for abstract level information processing and provides necessary tools for the task of knowledge-based Computer Aided Supervisory System Design - CASSD. The object oriented paradigm allows efficient representation of qualitative knowledge at various levels of abstraction and knowledge encapsulation becomes convenient and technically sound tool in the design of supervisory systems. Incorporation of qualitative and symbolic knowledge representation tools and rule-based system processing, in a single framework constitutes a new quality in the area of Computer Aided Control System Design and provides a user friendly, flexible and very powerful tool for design tasks covering both numerical and symbolic knowledge representation.*

The object approach forces engineers to structure knowledge becoming highly locatable, modular and encapsulated. This features are very important to get expert supervisory system design closer to process. The objective is to approach design tools to process engineers avoiding extra-time in learning application functionality and interfacing process variables and design tools. Thus, objects are used in the process variables descriptions as sources of information, encapsulating tools to provide significant (qualitative or numerical) information. Object oriented features will permit to divide large KBs into smaller ones to deal with complex systems adopting distributed solutions. Consequently, ES becomes more specialised, maintainable, and easier to validate.

Furthermore, in order to base the thesis on a stronger background a wide comprehensive review of domain knowledge and literature and practical solution is undertaken. All the proposed solutions are analysed in detail, implemented and tested. Examples of practical applications are provided. Finally, concluding remarks are presented.

1.5. Conclusions.

Domain of expert supervision has been briefly introduced to focus the problematic of using expert knowledge in the design of supervisory systems. The main problems are the dealing with heterogeneous types of information (numerical, qualitative, logic, knowledge, etc.). The complexity in using multiple tools, from different domains, for knowledge representation and data

manipulation becomes an integration problem. The use of a framework especially conceived to assist these tasks, falls within the scope of this thesis. Integration of AI based tools with classical numerical methods by means of an object oriented approach is proposed as a solution to encapsulate information coming from process and engineers at several abstraction levels.

The necessity of this CASSD (acronym of Computer Aided Supervisory Systems Design) framework is to support the iterative procedure involved in the development of such applications in all their steps (design, test, validation). A CASSD framework is needed to incorporate both, numerical (modelling and simulation, signal processing, analysis and representation) and knowledge-based facilities (representation and processing). Numerical to qualitative interfaces are also needed to deal with process variables at several abstraction levels.

2.

Expert Supervision

2.1. Introduction.

Nowadays, the interest for supervision is increasing due to the growing demands for quality, safety, reliability, availability and cost efficiency in industrial processes. As systems grow in size and complexity, the possibility of misbehaviour increases. Thus, the call for fault tolerant systems is gaining more and more importance. Fault tolerance could be achieved either by passive or active techniques [Frank and Köppen-Seliger, 1995] :

- The *passive approach* makes use of robust control techniques to ensure that the closed-loop system becomes insensitive with respect to faults. This solution allows small faults be tolerated without control system reconfiguration.
- The *active approach* provides fault accommodation, i.e., the reconfiguration of the control system when a fault has occurred. Reconfiguration can be thought at various degrees, i.e. set point changes, parameters re-tuning or structural changes. The aim of this approach is to avoid a fast degradation of the whole system due to this fault. The majority of actual solutions involve human decision.

In this work, only active approach is taken into account. This chapter overviews different methodologies that can be applied in this sense to remark the importance of human knowledge about process behaviour and how it can be used to implement supervisory structures. Finally, it is concluded that the use of both, analytical and knowledge-based methods, together can improve the results of supervisory structures. In such cases, the main problem is about integration of methods provoked by differences in data representation (numeric, qualitative, symbolic).

2.2. Supervision, fault detection and diagnosis : Terminology and definitions.

The terminology used in the literature in the field of supervision fault detection and diagnosis is not unique. Consequently, the Technical Committee SAFEPROCESS tried to find commonly accepted definitions. Some of these preliminary proposals are collected in [Isermann and Ballé, 1996]. The terminology used in this text has been transcribed when used with a similar meaning, and redefined when used with a different significance (such topics are marked with an asterisk, *) :

About states and signals :

- *Fault* : Unpermitted deviation of at least one characteristic property or variable of the system.
- *Malfunction* : Irregularity in fulfilment of a systems desired function.
- *Error* : Deviation between a measured or computed value of an output variable and the specified or theoretically correct value.
- *Disturbance* : An unknown (unmeasurable and uncontrolled) input acting on a system.
- *Perturbation* : An input acting on a system which results in a temporary departure from steady state.
- *Residual* : Fault indicator, based on model equations.
- *Symptom** : Change of the observed behaviour with respect to the normal one.

About functions:

- *Fault detection* : Determination of faults present in a system.
- *Fault isolation* : Determination of kind, location and time of detection of a fault. Follows fault detection.
- *Fault diagnosis** : Determination of the origin of a fault. Therefore, it follows fault detection.

- *Monitoring* : A continuous real-time task of determining the condition of a physical system.
- *Supervision* : Monitoring a physical system and taking appropriate action to maintain the operation in the case of faults.
- *Protection* : Means by which a potentially dangerous behaviour of the system is suppressed if possible or, means by which the consequences of a dangerous behaviour are avoided.

Although all of these topics exist in the bibliography and correspond to different stages in the study of faults of plants, the majority of works in the domain are centred on: fault detection, fault diagnosis, monitoring and supervision.

About models :

- *Quantitative model* : Use of static and dynamic relations among system variables and parameters in order to describe systems behaviour in quantitative mathematical terms (also called analytical or numerical model).
- *Qualitative model* : use of static and dynamic relations among system variables and parameters expressed in symbolic terms in order to describe systems behaviour in qualitative terms.
- *Diagnostic model* : A set of static and dynamic relations which link specific input variables -the symptoms- to specific output variables- the faults.
- *Analytical redundancy* : Use of two or more, but not necessarily identical ways, to determine a variable where one way uses a mathematical process model in analytical form.

About system properties and its measures:

- *Reliability* : Ability of a system to perform a required function under stated conditions, within a given scope, during a given period of time. It can be expressed by the Mean Time Between Failure (MTBF). It is the mean value of time passed between two consecutive failures
- *Safety*: Ability of a system not to cause a danger for persons or equipment or environment.
- Other terms such as, *availability or dependability*, are less frequent terminology, referring to probability of satisfactory operation of systems through time. They are not used in this text.

The scope of supervision is not only to detect malfunctions and faults, but also to propose actions against these situations. Therefore, basic tasks associated to a supervisory system have a correspondence with fault diagnosis, [Gentil, 1996],

and other fault related tasks. Once faults are detected and localised, actions can be proposed or ordered to assure global performances. See Fig. 2.1 for the relationship between tasks and terminology.

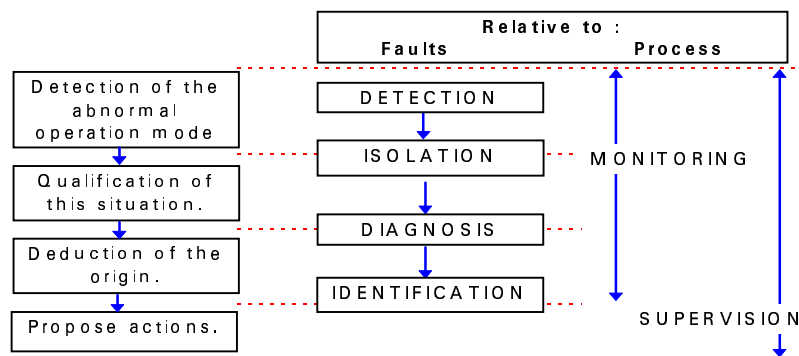


Fig. 2.1 Supervision tasks.

Nowadays, commercial industrial applications cover simple monitoring tasks that consist in data management (storing, visualisation and representation) and alarm generation. This is the case of extended SCADA packages. More advanced systems can diagnose and propose actions, but final decision about alarm certainty or action validity are restricted to human operators.

2.3. Fault detection: structures and methodologies.

Supervision is essentially the set of techniques used with the goal of assuring the integrity of a system. The definition given in the last paragraph assigns to supervision the role of detecting (to recognise and to indicate) in real time abnormal behaviour of a process taken benefit of all information available about the process (measures, models, history, experience and so on).

According to these goals, the main part of supervision of a complex system is focused on to *detect* and *isolate* occurring faults and *provide* information about their size and source, [Frank and Köppen-Seliger, 1995]. The most important and difficult task is centred on fault detection and diagnosis where difficulty increases with the real time constraints and complexity of systems (non-linear, coupled dynamics, time dependencies, etc.). The core of the fault diagnosis methodology is the so-called model-based approach, where either analytical or knowledge-based models or combination of both are used in combination with analytical or heuristic reasoning [Frank and Köppen-Seliger, 1995]. The classical procedure of a fault diagnosis system is depicted in Fig. 2.2. This is achieved in three basic steps, residual generation, evaluation and analysis, not always clearly separable.

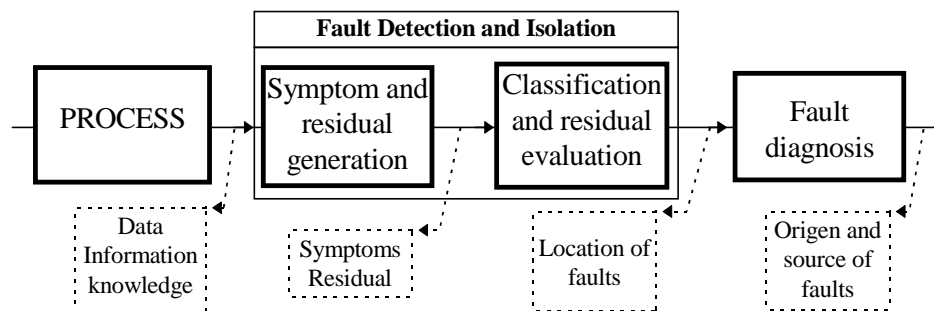


Fig. 2.2 Schematic representation of the procedure of fault diagnosis.

Hence, the methodology used in fault detection is clearly dependent on the process and the sort of available information. [Pal, 1995] gives an extensive classification and description of these methods (Fig. 2.3). A distribution of fault detection methods depending on applications is summarised in [Isermann and Ballé, 1996], from the main contributions presented in the domain main conferences from 1991 till 1995. The evaluation of fault diagnosis and supervision methods is more difficult because of little data, although rule-based reasoning methods are increasingly used and also the number of fuzzy rule-based applications is growing [Isermann and Ballé, 1996].

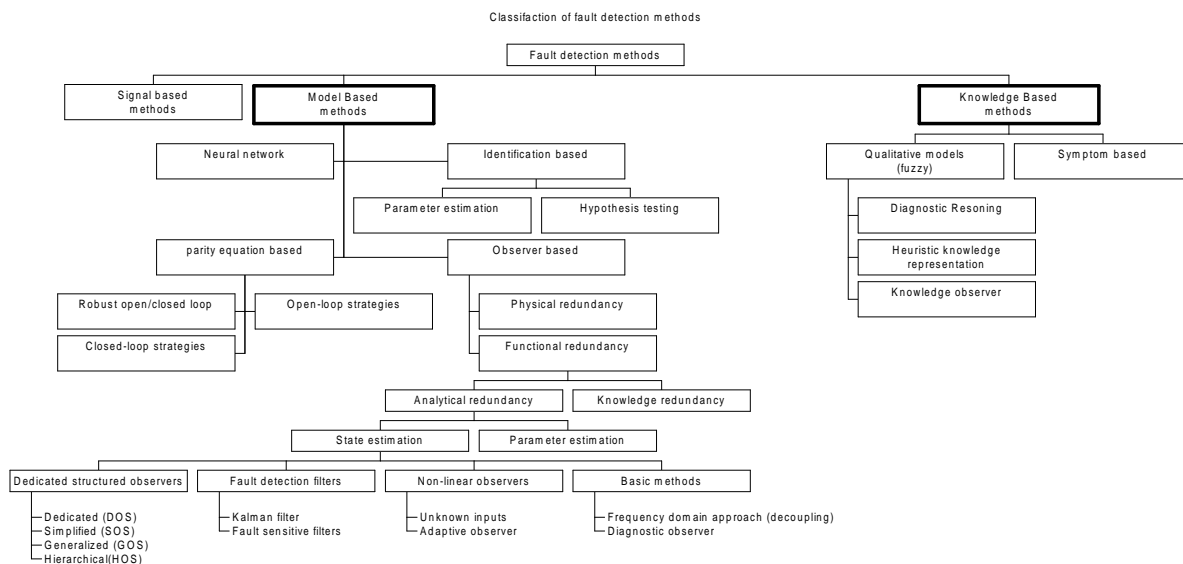


Fig. 2.3 Classification of fault detection methods [Pal, 1995].

Classification in Fig. 2.3 shows the diversity of methods available for fault detection using analytical models and the shortage of methodologies described in the case of the knowledge-based approach. The reason of this difference is due to the tools used in both situations. Most of the model-based fault detection

methods are based on comparing measures and simulations for obtaining a residual as is depicted in Fig. 2.4. The application of existing signal processing and statistical methods to deal with numerical data boosted the used of such methods. The major inconvenient of those methods is the necessity of a model. The knowledge-based model offers an alternative to that situation in which an accurate model is difficult to be obtained.

In the following subsections, fault detection methods are briefly described and classified in model, signal and knowledge-based methods. Model-based methods group analytical methods based on comparing process output with theoretical model response to the same input applied real process. The next subsection, signal based methods, is restricted to those methods that only use process output variables in the fault detection task and no model is used. Finally, the knowledge-based method subsection, introduces the problematic of dealing with knowledge representation to monitor and supervise dynamic systems.

2.3.1. Model-based fault detection.

Model-based fault detection methods are focused on residual generation, i.e. fault indicators (Fig. 2.4). Residuals are obtained as changes or discrepancies in special features of the process obtained from process variables (for example, output signals, state variables) or coefficients (for example, estimated parameters or other calculated ratios). To achieve this goal, data obtained from the process is compared to the data supplied by models representing normal operating conditions. For this, different change detection methods are applied. The next step consist in residual evaluation to decide about faults existence. Threshold logic, statistical decision theory, pattern recognition, fuzzy decision making or neural networks are actual methods used to decide whether and where a fault has occurred.

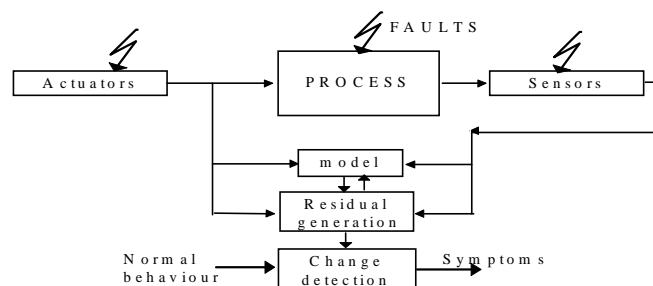


Fig. 2.4 Architecture of a model-based diagnostic system from [Maquin and Ragot, 1996].

Different kinds of process models and methods can be used to generate residuals of state or output variables. A basic classification of them is represented

in Fig. 2.5. Observer-based residual generation consists in using observers or Kalman filters to reconstruct the interest output of the system. Then, the error between real data and estimated data or a function of them are used as residual. In the parity space approach the process equations (for instance, state space equations) are modified with the aim of getting residuals decoupled from system states and different faults. The inconsistency of these parity equations represent the residuals. On the other hand parameter estimation methods are based on the assumption that the faults are provoked by changes in the physical system parameters (mass, friction, resistance, viscosity, etc.). Therefore, these methods use process measures to repeatedly estimate the parameters of the actual process. Estimations are compared with the parameters of the reference model, obtained under fault-free conditions. A wider description of those methods and other interesting variants of them are included in [Frank, 1996].

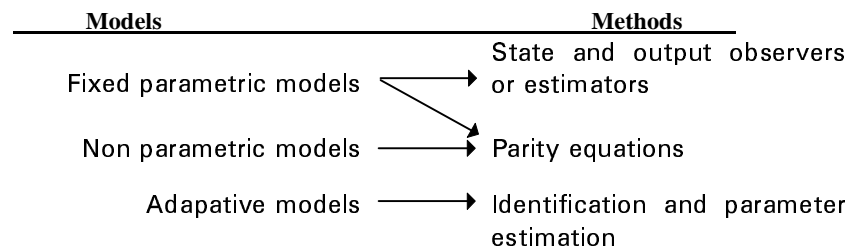


Fig. 2.5 Dependency between method and model to be used in a model-based fault detection.

Although these approaches have proved to be very effective in many applications, they have two major shortcomings : the complex technology or natural process are generally non-linear time-varying systems, which makes it particularly difficult to detect structural changes in the system and to obtain adequate models for this purpose. Secondly, the available model is often assumed to represent normal operating conditions, and the impact of a departure from these conditions on the model outputs is difficult to predict [Du, Elbastawi and Wu, 1995a]. Consequently, these methods are difficult to be applied to dynamic process submitted to repeated changes in the operation mode.

2.3.2. Signal-based fault detection.

Model-based fault detection requires process variables (measures) to compare real process response and model response. This comparison is performed under the assumption that the same input is provided to both systems. Therefore, process measures and actions must also be supplied as input of model. Other methods can be applied if only process output is available, i.e. signal-based methods. This methods are usually used with rotating machinery and electrical circuits and applied with signals measured from process in steady state.

Therefore, signals are thought to be rich in information. This is the case of vibrations analysis and other methods based on frequency analysis.

Signal-based methods are focused on analysing signal features. Change detection is measured as a deviation from normal behaviour. For this purpose, statistical (mean, variance, entropy, etc. are estimated), frequencial (as filtering or spectral estimations methods for example) and probabilistic (Bayes decision) methods are used. Signal models or patterns are used to detect deviations from normal operating modes. Those methods do not require the mathematical model of process. Knowledge about system is assumed to consist in learning associations between process measures and operating conditions. In this sense they can be considered as knowledge-based methods.

Some of the limitations of pattern recognition techniques is that they assume a knowledge about all systems states and do not take into account the time evolution of the process under study. A survey of these techniques can be consulted in [Denoeux, Masson and Debuissou, 1996].

2.3.3. Knowledge-based fault detection.

In the case of noticeable modelling uncertainty, a more suitable strategy is that of using knowledge-based techniques. Instead of output signals any kind of symptoms can be used and the robustness can be attained by restricting to only those symptoms that are not strongly dependent upon the systems uncertainty. In this case, knowledge has to be processed which is commonly incomplete and can not be represented by analytical models. On the other hand, residual evaluation is a complex logical process which demands intelligent decision making techniques, like fault tracing in fault trees or Petri nets or pattern recognition including fuzzy or neural techniques. Therefore, knowledge-based methods are quite a natural approach also for residual generation in fault diagnosis, and ESs have so far been applied more successfully here than in the field of control ([Frank, 1996]). The use of knowledge, in the model definition or qualitative observation of variables, when analytical models are difficult to be obtained, is another field where AI techniques can be used.

Knowledge-based methods is a field in continuous evolution, where AI techniques have an important role. There is not a unified theory to be applied to these methods and, in fact, knowledge-based methods can be applied in all three phases of fault diagnosis, namely residual generation, residual evaluation and fault analysis, although the phases in this case are not always as clearly separable as in case of the analytical approach. [Frank, 1996] distinguishes two categories

in the knowledge-based domain for diagnosis, also applied for residual generation:

- *Symptom based.* It consists in organising expert knowledge into diagnosis ESs. Then, ESs deal with process variables to identify fault symptoms in them (See Fig. 2.7). When symptoms are considered in connection with process input, we speak of a *symptom-model-based* approach. In such implementations, major difficulties exist in the knowledge acquisition task and knowledge representation, for obtaining a rule base for example, and knowledge processing and interfacing, when running the ES application with real data.
- *Qualitative model-based.* This methodology is based on using process knowledge to represent systems structure in terms of rules and facts. The goal is to dispose of a rough model to be used as a model-based approach. The use of qualitative techniques implies a description of process variables given by short sets of labels or symbols (low, normal, high). Consequently, the inevitable deviations from the exact model, (uncertainties and incompleteness) always present in these representations, requires the AI support.

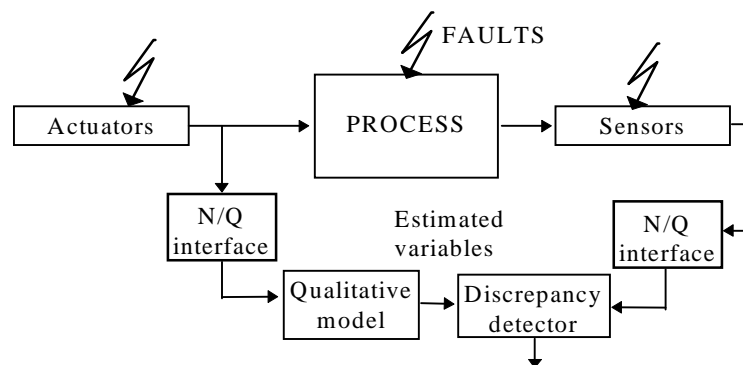


Fig. 2.6 Basic scheme of qualitative model-based fault detection method

The use of one or the other method is only submitted to the knowledge about process behaviour or faults. Furthermore, there are not exclusive methodologies and a combination of both can be required. In fact, the best strategy tries to use all available knowledge and data for reaching the fault detection goal.

Additional problems, when dealing with knowledge representation, of process behaviour or faults description, occur when interfacing fault detection structure (linguistic representation of magnitudes and process variables) and process (numerical variables, measures). However, the use of all available information, i.e. numerical data and knowledge, can improve fault detection structures because they are complementary approaches.

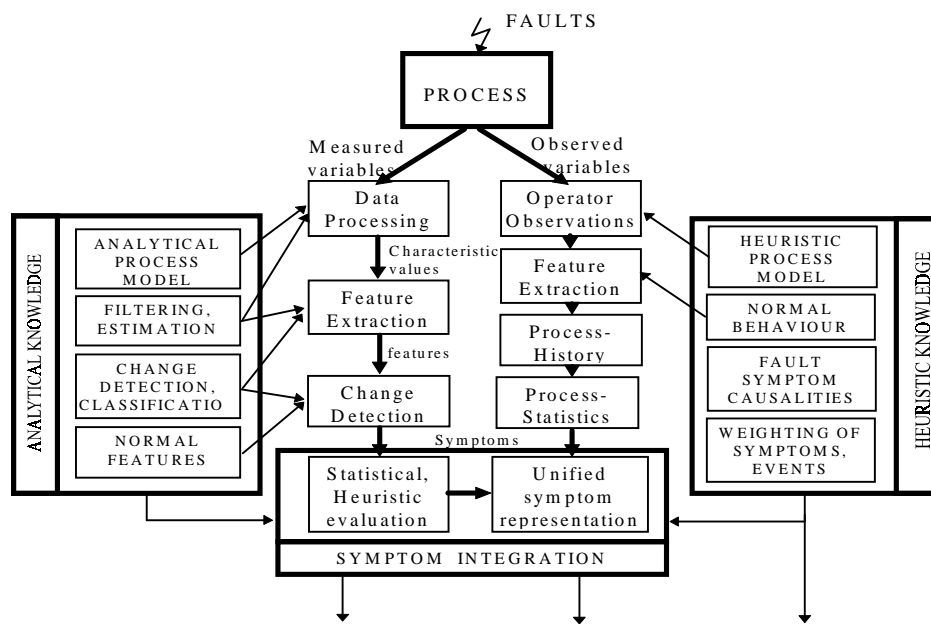


Fig. 2.7 Scheme of knowledge-based fault detection based on symptom generation (From [Isermann and Ballé, 1996])

Fig. 2.7 represents how both, numerical techniques and knowledge-based techniques, can be merged for fault detection. In fact, heuristic knowledge acquired from process observations is used in this proposal to reinforce analytical methods applied to numerical variables in the symptoms generation step. While numerical methods are used for features extraction of measured variables, i.e. signal processing techniques as filtering and analytical estimations, some drawbacks appear in extracting features from operator observations. Specialised knowledge processing techniques must be used with this aim. Taking into account that difficulties in such systems are presented at knowledge representation level, the use of those techniques for features extracting are not very extended.

2.4. Knowledge-based fault diagnosis.

Fault diagnosis follows fault detection in the supervisory chain. According to the general description of fault detection systems, residual evaluation is a step preceding fault diagnosis. Constraints and conditions used in fault evaluation are submitted to particular faults. In these tasks, the use of intelligent decision systems offers a clear advantage because the use of expert knowledge about fault in the decision conditions. The use of AI techniques in this task can be combined with the analytical model used for fault detection.

2.4.1. Knowledge-based methods used in fault diagnosis.

The extensive domain of AI can be applied in different ways to diagnose about faults. In this sense, logical decisions needed in residual evaluation, can be performed by using fuzzy logic. Adaptive threshold for uncertain systems can be performed by fuzzy rules. In such situations, rules incorporate knowledge about process behaviour and the threshold is changed according to the operation conditions.

Another powerful technique from the AI domain are the Artificial Neural Networks. They have also been used in fault diagnosis with different purposes such as residual generation by replacing analytical models by trained networks and using them as a normal operating model. Residual evaluation can be improved by using neural networks using data from previous faults to train neural networks to solve these situations.

Neural networks or fuzzy logic are AI technologies that can be used within analytical model-based fault detection for improving them. In fact, both neural networks and fuzzy logic use numerical data. Another different approach consists in using qualitative models (process models or faults models) for diagnosis as it has been introduced when talking about knowledge-based fault detection. In the case of a fault model, a previous knowledge about faults symptoms is assumed, then process is observed to match some of them. On the other hand, process-model is used to detect qualitative deviations from the normal operating mode.

Partial work of this thesis has been focused on the task of developing and integrating tools to assist knowledge-based design using the two last approaches. Main effort has been done in providing a useful tool for representing the simple qualitative relationship, ALCMEN, to perform qualitative observations of process variables submitted to faults. A case example has been developed to test both techniques. The use of ALCMEN to qualitatively estimate a non-measured variable is described in [Melendez et al. 1996a], while the analysis of measured process variables in fault situations is used to obtain a qualitative description of faults in [Melendez et al. 1995]. In this case the use of rule-based ES under G2 was used to diagnose. The use of imprecise description of signals features, given by qualitative labels, as antecedents provoked sudden transitions in the ES decision because the ES was not able to manage this imprecision. This work was improved, by using a shell with fuzzy reasoning to deal with uncertainty and imprecision [Sàbat, 1996].

Sometimes the necessity of applying one of two methodologies, symptom-based or qualitative-model-based, is not clear and knowledge about fault

situations and qualitative estimations of variables can be merged to improve the diagnose system. Moreover, it is necessary to combine both methodologies to take benefit of KB describing symptoms related to non measured variables, i.e. the qualitative observer. Former work of the author, with the same case example, has been extended in this sense as is explained in [Melendez et al., 1996b].

2.4.2. Diagnosis strategies.

Despite there does not exist a standard architecture for fault diagnosis, actual tendencies point to hybrid architectures because of the benefits of merging both technologies, (analytical and model-based approaches). Basic and pure lines are represented in Fig. 2.8 but drawbacks are presented in implementing simple strategies, due to the incompleteness of information when dealing with any proposal, especially with complex systems.

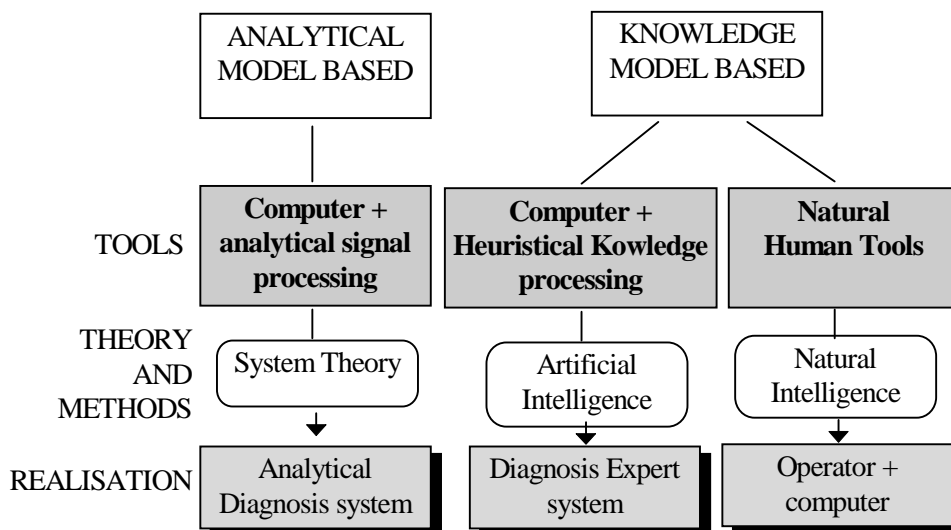


Fig. 2.8 Strategies for fault diagnosis

Supervision of complex systems (non-linear, time dependent, etc.) becomes impossible using analytical model-based approach because difficulty in obtaining accurate models. Despite of this drawback numerical methods must be taken into account for obtaining signal features that are needed by ES, representing expert knowledge, to deduce about behaviour of process variables. This thesis is particularly focused on combining all possible techniques in the implementation of complete supervisory systems. Fig. 2.7 represents the variety of information that can be used in the implementation of knowledge-based structures for fault detection and diagnosis.

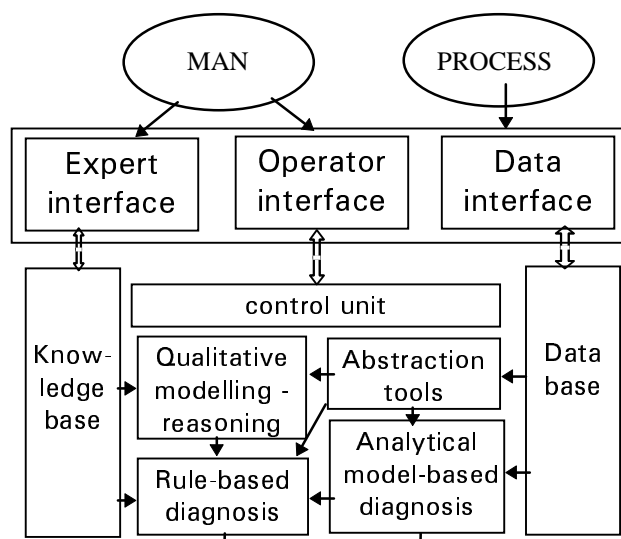


Fig. 2.9 Hybrid, Knowledge and analytical based architecture for fault diagnosis

In this ways, the proposal of this thesis (Fig. 2.9) tries to take benefit of both knowledge and analytical tools, providing abstraction tools as numerical to qualitative interfaces. Qualitative reasoning and modelling are also present to roughly estimate process variables. Rule-based systems must be used together with analytical tools in the diagnosis tasks because data coming from process is purely numerical and process knowledge is referred to qualitative perceptions of them. In following chapters a set of tools are selected to be integrated in a commercial framework to facilitate the development of supervisory structures following this line.

2.5. Implementation of expert supervisory systems.

Despite implementation of supervisory systems is not an extended topic in literature, it is an important stage. Especially, when dealing with knowledge-based methods because of the limitations of available shells. Nowadays, knowledge-based methods are thought to be useful in all supervisory tasks. The easy use of rule-based ES for classifying or the use of graphs or fault trees as analysis tool, are simple tasks where expert knowledge can improve fault detection. Moreover, the use of fuzzy logic to deal with imprecision and uncertainty, or neural nets for classification, are other representative examples of how AI is coming into supervisory schemes.

Nowadays, final actions are restricted to human decision, then the implementation of intelligent supervisory systems is far from the actual implementations in industrial process and restricted to simple controller

reconfigurations or rule-based controllers (fuzzy controllers). A reason for these difficulties in implementing expert supervisory systems is the complexity in knowledge representation and validation of such applications.

2.5.1. Assisting expert supervision design.

A possible block representation of supervisory system, is depicted in Fig. 2.10. A model-based (analytical or qualitative) fault diagnosis system is used to reconfigure the control system. Actual shells and existent frameworks used for industrial monitoring tasks (SCADA systems as CITECT, InTouch and so on.) permit such kind of reconfiguration according simple conditions. These are well dotted of numerical methods and tools, but they lack of knowledge-based methods. In fact, the application domain of these applications is reduced to simple monitoring, representing, alarm generation and registration tasks. On the other hand, actual shells conceived to deal with knowledge-based systems are designed to manage specific kind of knowledge representation and there exist some difficulties in integrating numerical capabilities. This is the example of the shell G2, the actual state-of-the-art in real-time knowledge-based process diagnose tasks. It is provided with an object-oriented graphical user interface that offers several knowledge representations tools, such as rule bases, frames, tables. G2 inference engine is able to deal with KB in several ways (forward and backward chaining, focusing, invoking). Despite of its capabilities of knowledge representation, it presents some drawbacks in qualitative modelling and uncertainty (and imprecision) management, which are necessary in a complete knowledge-based framework. Another inconvenient is detected when dealing with multi-level representation of process variables. In such cases it is necessary to abstract significant information from numerical measures, and those frameworks do not provide sufficient tools for numerical management and signal processing.

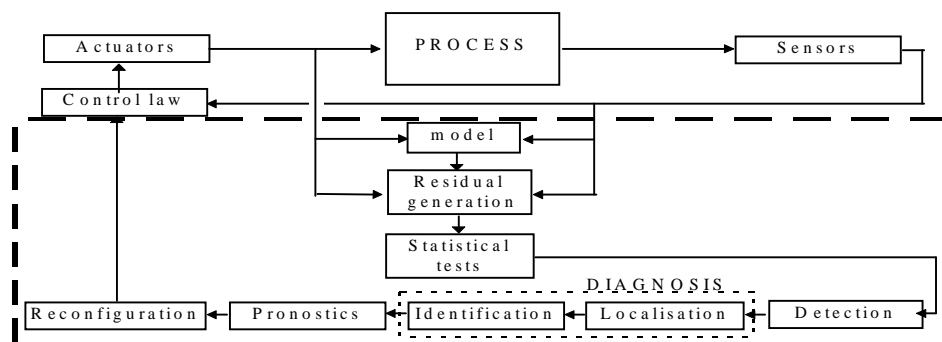


Fig. 2.10 Model-based supervisory scheme.

Another important factor to be taken into account in the design of supervisory systems is the application domain. Different tools and signals must be used for process supervision than for controllers supervision. It also applies to continuous, discrete or hybrid systems and to distributed or centralised systems.

Management of both numerical and qualitative data and the use of analytical and knowledge-based models and tools is necessary in actual supervisory systems. Taken into account that the supervisory loop is closed by means of reconfiguration of the control systems, a supervision shell must also include control systems design capabilities and some evaluation and graphical representation tools. The actual Computer Aided Control Systems Design (CACSD) frameworks incorporate such capabilities and a great number of numerical algorithms, including modelling and simulation capabilities. On the other hand, these frameworks are not provided with knowledge-based capabilities. Consequently, a possible approach for assisting expert supervisory systems design based on control reconfiguration, is to add following knowledge-based capabilities to CACSD frameworks:

- *Numeric/Qualitative interfaces.* Process variables are the main link between the process and the supervisory system. Therefore, different observations of them must be possible and, consequently, a multi-level representation of them must be performed. Numeric features of process measures, as its actual value, derivatives, mean or trend estimation, must be available together with qualitative representations such as qualitative tendency, labels or landmarks, to be useful for working together with both analytical and knowledge-based supervisory methods. Then a set of different methods for obtaining accurate qualitative representation of signals are needed to be used as numeric to qualitative interfaces. i.e. abstraction methods. The object oriented approach can be used to encapsulate the multi-level representation of these variables (*object-variables*) as it is explained in chapter 5.
- *Knowledge representation.* Several tools can be used to deal with expert knowledge at several abstraction degrees. Knowledge about process can be about process variables, the relationship between them, the description of situations (faults) related to physical elements, functionality, behaviour and so on. All of these possibilities require some specific tools with different representation capabilities.
- *Knowledge processing.* Usefulness of knowledge-based systems reside in the capability of deduction using knowledge representation. Then those inference engines must also be present in the framework to deal with qualitative relationship (*qualitative reasoning*), rules (ES), logic, and other possible knowledge representations. Capabilities of certainty and imprecision management must also be present, for example, by using fuzzy reasoning.

Some tools are selected and presented in chapter five to assist in the design of control systems supervision. With this goal a specific commercial open package from the control domain, MATLAB/Simulink, has been used as a platform where knowledge representation capabilities and qualitative reasoning tools have been integrated by means of an object oriented approach [Melendez et al., 1996b]. The goal is to take advantage of the existent representation and analysis tools used in control systems design, extending its capabilities with knowledge-based techniques to assist engineers in such designs, avoiding the always difficult problem of interfacing separated tools reasoning on dynamic systems. The work has been centred on solving main drawbacks in knowledge representation and processing. With this aim, a tool for dealing with simple qualitative representations, ALCMEN, and an ES shell, CEES, have been added. Other possibility is pointed by [Rengasamy, 1995], using CIM models to integrate several functionalities in order to assist supervision.

2.6. Conclusions.

Supervisory systems design is an actual research line that covers process monitoring at several stages (fault detection, diagnose, and identification). This chapter has been focused on the necessities of knowledge-based systems in the different steps of supervisory systems design.

Nowadays, the main research activity in the area of supervision is focused on the objective of fault detection. Several methods are described in the literature, but there does not exist an unified theory. Model-based theory has been consolidated for numerical models despite of the necessity of an extension for non-linear systems. In the case of knowledge-based fault detection, qualitative models can be used to roughly deduce non observable dynamics and symptom description by means of rule-based systems, can use this information. Nowadays, qualitative models implementation is supported by several AI technologies.

The combination of both analytical and knowledge-based methods in fault diagnosis falls within the field of the actual research and in the scope of this work. Numeric to qualitative and qualitative to numeric interfaces, knowledge representation and processing are relevant topics to be solved before a unified approach could be reached. When dealing with complex systems (large systems, non-linear, time dependent, coupling and so on), main difficulties appear in the implementation of global supervisory systems because of the amount of different data and information to manage. The global conclusion is that there does not exist a framework to deal with different approaches (knowledge and analytical based) in the design of supervisory systems.

3.

Methods for Expert Supervisory Systems

3.1. Introduction: Necessity of AI methods for process supervision.

The complexity of designing supervisory systems has been introduced in the last chapter. The necessity of incorporating expert knowledge in such design is present in all tasks involved in supervisory systems. Fault detection can be performed by using analytical models, but these models are not always available and final resolution about residual generated, is always submitted to the human expert decision. On the other hand, the use of knowledge-based representations is needed by numeric to qualitative interfaces to perform reasoning about process variables. The important role of human knowledge in process supervision is concentrating the attention on these techniques that permit the use of expert knowledge to automate these tasks. From this perspective, the increasing use of

Artificial Intelligence (AI) techniques in control and process engineering must improve supervisory designs, and coexist with numerical methods.

The most extended AI tools in the control domain, are overviewed in this chapter, and benefits and drawbacks of using some of the AI techniques for knowledge representation and qualitative reasoning, are discussed from a supervisory systems design point of view. The aim is to take advantage of control experience in using AI resources, to choose a set of adequate tools to be integrated in a framework for assisting supervisory tasks avoiding.

3.1.1. Heterogeneous, imprecise and uncertain data.

In the design of supervisory systems, the information related to process variables is available in several ways: numerical data (from sensors, analytical models, numerical estimations and so on), qualitative data (from human perception of process variable trends, qualitative models and qualitative estimations) and relationship or dependencies among these data. The complexity in managing together such different kinds of data is in the scope of AI techniques.

Nowadays, the main contribution for designing supervisory systems comes from operators and expert engineers experience. In fact, they are also present in the majority of applications for final decision making. Sometimes, the description and translation into computers of this expert knowledge, related to process variables, becomes very difficult or impossible due to the different nature of human descriptions and data obtained from process.

Usually, experts describe situations, while data are instantaneous samples of measures, or estimations of these situations. In the procedure of matching process variables evolution and those situations, humans use an imprecise description of magnitudes. An example can clarify these difficulties : The following sentence, “when *temperature* in the reactor *increases*, *opens* the input *valve slowly*”, describes an action (*opens valve slowly*) to be performed when a process variable (*temperature*) experiments certain behaviour (*increasing*). This expert description are easily interpreted by humans, but difficult to interface with numerical magnitudes coming from the process (*temperature*) or actuators to perform this action (*open valve*). They are imprecise descriptions of numerical magnitudes available in the process. This imprecise description must be processed before to be used in the control structure. The representation of these kinds of information and the capability of dealing with the relationship among imprecise variables is in the scope of AI. The use of qualitative reasoning and modelling techniques can be useful for these purposes.

Another important feature of AI techniques is the capability of dealing with uncertain data or information. The description of an intermittent fault is an example of this situation. In spite of some conditions matched, the fault is not sure. Certainty can be introduced as an index of confidence in the description of situations or data. A similar situation is given when contrasting information among several sources. Sometimes it leads to incongruous descriptions and a decreasing confidence of a source of information. In such cases, the use of different certainty indices for each source can be useful to merge incoming information to work in co-operation. See [De la Rosa J.LI, 1994].

3.1.2. Process behaviour and expert knowledge.

An additional inconvenience of expert knowledge for process supervision refers to temporal references. Usually it describes process behaviour in an uncertain period of time, or changes in the evolution of process variables without dating these events. In the example of the previous paragraph, the label *increasing* is related to a characteristic of a process variable during an imprecise period of time. This consideration must be taken into account for building numeric to qualitative interfaces in order to take benefit of this kind of descriptions of variables evolution. This also applies more general descriptions of process behaviour such as *transient* or *steady state*, for instance. In this case, both possibilities are exclusive, but real transition between both states is gradual. Then difficulties exist in determining the limits between both, because it implies all process variables analysed.

Actual AI techniques, as fuzzy logic or some qualitative reasoning formalisms, are applied to deal with this imprecision in the description magnitudes and the relationship between variables. On the other hand, temporal imprecision is inherent to qualitative methods.

3.2. Knowledge representation techniques.

The main characteristic of AI techniques is the separation between knowledge and inference mechanisms. This independence causes that two separated steps must be performed to build any AI based system. The first is to declare knowledge and second is to reason about facts according to the previous declaration. Consequently, these tools are provided of mechanisms for knowledge representation and manipulation, and mechanisms for reasoning.

In order to encode expert knowledge concerning the process, different knowledge representation techniques could be used. The most popular ones include the following :

- Logical formalisms
- Rule-based systems
- Graphs (including semantic networks)
- Frames and object oriented representations

Of course, more complex systems use combinations of the above knowledge representation schemes. Below, a brief presentation of the core formalism mentioned above, is given. It is not in the scope of this work to start an extensive discussion of these techniques, but only to present the different possibilities for representing expert knowledge in supervisory applications.

3.2.1. Logical formalisms.

Logical formalism includes both propositional logic and predicate calculus. In the simplest case, propositional logic is satisfactory as a knowledge representation formalism. The language of propositional logic consists of propositional symbols, such as p, q, r, \dots , denoting logical sentences (a logical sentence is one to which we can assign the truth value, i.e. either *true* or *false*) and logical connectives. Among the logical connectives, typically used, are : \wedge (conjunction), \vee (disjunction), \Rightarrow (implication), \Leftrightarrow (equivalence) and \neg (negation). Using the above connectives and parentheses, various complex logical sentences can be build.

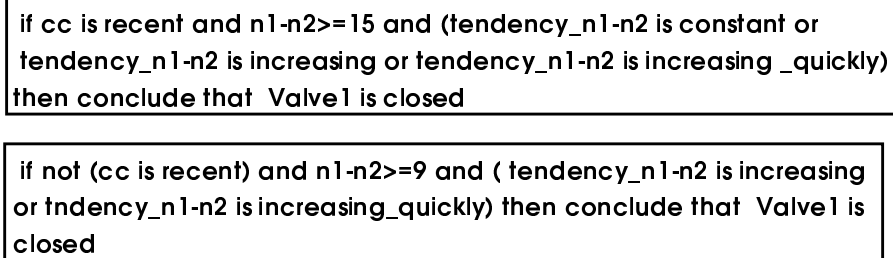
The main advantages of logical formalism include well established formal background and theoretical properties (soundness and completeness of inference systems). Further, in case of logical formalism, the standard logical inference methods are immediately applicable.

3.2.2. Rule-based systems.

Rule-based systems, are the most comprehensible form of knowledge representation. These are descriptions of condition-dependent actions. A single production rule is understood to be a single item of information. The simple structure “IF ... THEN ...” has become the most successful way of knowledge representation. Production rules applied in a specific domain by representing expert knowledge applied to a particular problem are also called, expert systems (ES).

The most important component of these knowledge representation techniques is the inference mechanism. The inference mechanism controls the selection and activation of production rules. This inference engine must be able to solve possible conflicts appeared in the selection procedure and multiple deductions of the same fact.

When using ES for representing knowledge involving numerical data, additional capabilities of processing uncertainty and/or imprecision are necessary, due to the imprecision of input data. In such situations, it is evident that the use of adequate numeric to qualitative interfaces will improve the results. To dispose of a set of correct rules is as important as to provide the ES with expected inputs in the antecedents of these rules. Therefore, when using ES directly connected to dynamic systems, the additional mechanism must be used to interface process and ES.



if cc is recent and $n1-n2 \geq 15$ and (tendency_n1-n2 is constant or tendency_n1-n2 is increasing or tendency_n1-n2 is increasing _quickly) then conclude that Valve1 is closed

if not (cc is recent) and $n1-n2 \geq 9$ and (tendency_n1-n2 is increasing or tendency_n1-n2 is increasing_quickly) then conclude that Valve1 is closed

Fig. 3.1 Example of rules in the shell G2, from [Melendez et al. 1995]

Rules description is different according to the shell to be used and reasoning capabilities used by the inference engine of the ES. For example, Fig. 3.1 and Fig. 3.2 show two different rules from two shells, G2 and CEES respectively, especially conceived to deal with data coming from dynamic systems. G2 is a commercial shell conceived to deal with knowledge-based applications in the domain of process control. CEES [De la Rosa J.LI, 1994], is a research tool in continuous evolution according to the supervisory systems necessities. Basic differences between the rules are given by the ES capabilities of dealing with imprecision and uncertainty. In both, a qualitative description of the process variables is given, but ES in Fig. 3.2, CEES, can deal with imprecision and uncertainty, while ES in Fig. 3.1 can not. The activation of rules in Fig. 3.1 depends, exclusively (apart from process dynamics), on the interface numeric to qualitative used, while in the second example, Fig. 3.2, this activation can be tuned according to the result of smoothed comparison -for instance, $(Tendency_{n1-n2}).fv.equal(increasing)$ - using fuzzy sets in the definition of qualitative values, and the election of an activation threshold. The necessity of reasoning about numerical data makes the use of a shell with capabilities of dealing with uncertainty and/or imprecision necessary.

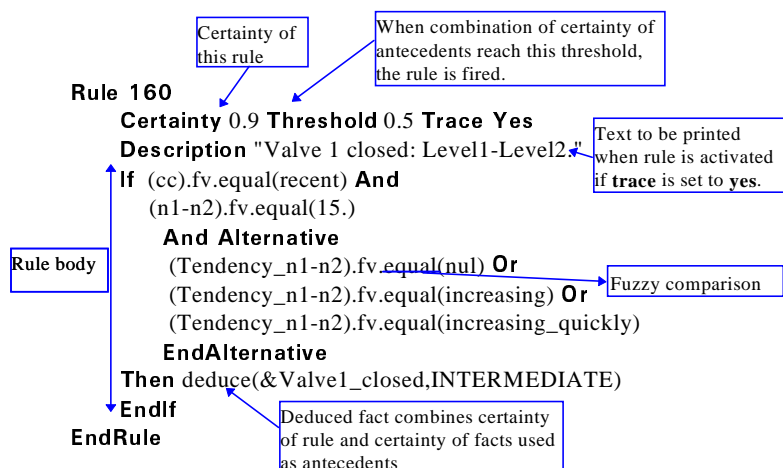


Fig. 3.2 Syntax of a rule in the shell CEES.

3.2.3. Graphs and including semantic networks.

Graphs and semantic networks are a means of representing knowledge-based on the relationships between objects. The nodes correspond to these objects, and links (lines or arcs) between them describe the dependencies. This is only a kind of representation without capabilities of processing. Final implementation of this kind of representation usually uses rule-based systems.

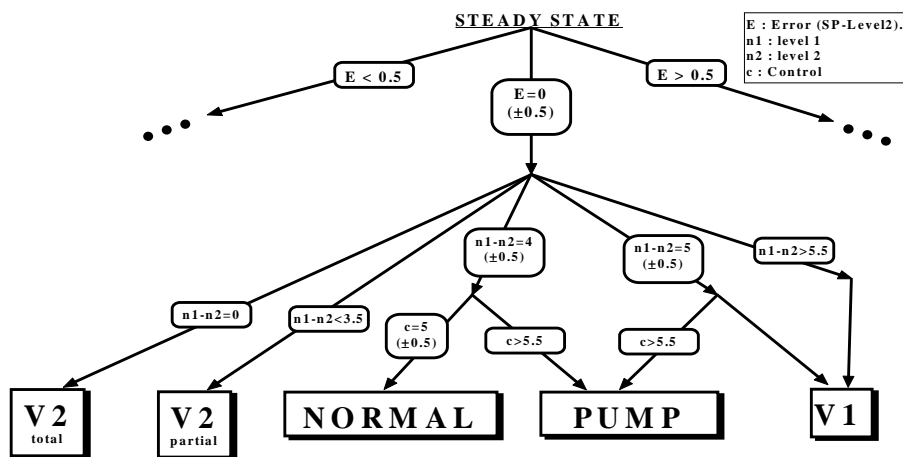


Fig. 3.3 Graph representation of process steady state situations.

Examples of Fig. 3.3 and Fig. 3.4, extracted from [Melendez, 1998], are a partial representation of the global process behaviour. In this case, graphs were used to assist the expert system configuration. The necessity of describing time dependencies to differentiate situations in the process behaviour, can be observed. These are the main difficulties when dealing with dynamic systems,

because the reasoning must be performed not only about sampled data, but also about the temporal evolution of signals (trends). The graph of Fig. 3.4 shows the necessity of supplying these trends ($E \downarrow$, or $E \uparrow$ for example) for interfacing coming data in the on-line application.

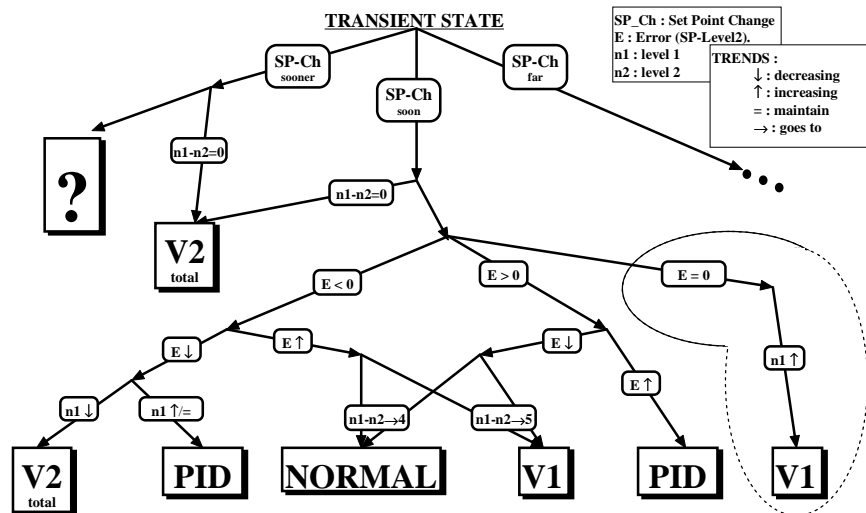


Fig. 3.4 Graph representation of dependency among process variables and faults in transient state.

In those cases, the imprecision of human knowledge is difficult to be interfaced with numeric process variables. In the previous graphs, the arrows are used to describe the human perception of increasing or decreasing tendencies of process variables in a sample interval. The translation of this observations into a rule-based system becomes a difficult interfacing problem. The use of specialised tools to abstract qualitative representation from process variables are needed to interface process input variables with expert knowledge.

3.2.4. Frames and object oriented representations.

A frame is a data structure for representing objects, situations, facts or other kind of knowledge by breaking it down into their constituent parts. A simple example of using frames consists in structuring data in tables. The use of frames involves a set of mechanisms (access, new, reset and change of values) to deal with frame *attributes* or *slots*. Inheritance is another important property of frames. Graphical representation of frames offers an easy to use interface to users. For instance, frames are present in G2 to represent knowledge related to process variables and objects. Attributes as sampling time, length, connections

and other are used to describe data and relationship between them. Copies of these graphical objects conserve all the attributes (inheritance).

The actual object oriented languages are very extended to support this type of representation with additional features. Languages such as C++ or Smalltalk permit to declare complex data structures, called *classes*, embedding data (object attributes) and methods (to manage and/or to access to its attributes). Main benefits of Object Oriented Programming (OOP) are resumed in the following properties :

- *Abstraction* : Once a class is defined, the instance of objects of this class is done with independence of the contents of this class. This allows to work in a more conceptual level.
- *Encapsulation* : This is the property of object oriented languages of storing data and methods in the same structure.
- *Polymorphism*, refers to the property of using the same description for an operator dealing with different kinds of data. For instance, an operator *addition* could be applied to different type of data (real, complex, integer) : *addition(real, real)*, *addition(integer, integer)*, *addition(complex, complex)* using the same name.
- *Inheritance* : Classes definition can take profit of the previous defined classes or partial properties without rewriting code.

The basic difference between OOP and procedural programming resides in that the data is not clearly separated from the code. The programming structure called *object* is used to encapsulate both, *data* (or *attributes*) and *methods* (procedures describing how actions of these objects must be performed) related to this data forming complex structures. In OOP, objects are created as instances of *classes*, or template structures, as in procedural languages variables are declared according to types of data. The extended use of object oriented languages in the domain of automatic control is surveyed in [Jobling et al. 1994].

3.3. Qualitative reasoning.

Qualitative reasoning (QR), also called qualitative physics, has become a domain of AI since the early eighties. This is the research area within AI which deals with human reasoning about physical systems [Bobrow, 1984]. Fundamentals of QR are based on applying human common sense and scientific implications used by engineers in the analyse of the environment and situations. The aim is to use a small number of symbols or qualitative values in the variables description. In dynamic process supervision, QR is used for describing qualitative

dependencies or models between symbols representing process variables. Two fundamental principles must be taken into account for qualitative reasoning:

- Continuous variables granularity is defined according to the type of reasoning to perform. For example, temperature qualitative values for alarm generation could take two states (*normal*, *high*) to differentiate a normal operation condition from a dangerous situation. The same variable used in a control loop could be defined using five qualitative values (*very low*, *low*, *normal*, *high* and *very high*) to smooth big and sudden transitions in the control law.
- Qualitative data evolution is event driven. This means that qualitative data is not equi-sampled and its changes in its values are asynchronous. From the previous example, temperature changes from *normal* to *high* only when the risk situation is approaching.

The intent of translating numerical models or relationships into qualitative descriptions involves the use of mathematical operators and calculating mechanisms defined for symbolic values. In this sense, qualitative representation has serious restrictions and algebraic operators are only defined for simple qualitative representations of numerical spaces. Consequently, the majority of actual pure qualitative mechanisms operate with qualitative variables that can only take the values : negative, zero or positive ($\{-,0,+ \}$).

For the development of this work, special interest for QR is centred on the use for developing simple qualitative observer. This is for describing simple relationships between qualitative variables obtained from process variables. The goal is to deduce the dynamic of variables that are not available as numerical data. For this purpose, other numerical variables can be used, using a qualitative description of them and the adequate relationship.

Other important QR formalisms are introduced in the next chapter as *abstraction tools* because of their capabilities in providing qualitative interpretations of numerical information at several degrees of abstraction. On the other hand, other QR formalism, such as algebra of signs and order of magnitude, are not present in this text due to the imprecision obtained in the results when multiple operations are combined (and multiple steps must be performed) in the deductions.

3.3.1. Qualitative modelling and simulation.

Modelling has been one of the areas where qualitative reasoning has been more successfully applied. Several qualitative formalisms succeed for qualitative modelling of dynamic systems. Various QR methods about physical systems are

based on the description of the functional mechanism. Then, using simple descriptions of process dynamics, qualitative simulations can be performed to determine the system behaviour from an initial condition. Some examples of techniques used with this purpose are :

- Envisionment [De Kleer and Brown, 1984] is a behavioural description consisting in a transition graph where all possible evolution of the system from an initial condition are described by the output branches. Model is constrained to the symbols used to represent physical parameters of the system, these parameters are interrelated by means of equations.
- QSIM [Kuipers, 1986], is a package conceived to deal with qualitative simulation. This approach differs from the previous in the definitions of qualitative values. In QSIM qualitative values are defined by a pair $\langle qval, qdir \rangle$, where $qval$ is obtained from an ordered set of landmarks and the intervals between consecutive landmarks, and $qdir$ is the qualitative derivative associated to this variable. Possible values for inc are dec (*decreasing*), std (*steady*) or inc (*increasing*). A set of constraining equations are composed by a variety of primitives (arithmetic, functional, derivative). QSIM has been improved, by integrating quantitative information, to decrease the ambiguity of managing pure qualitative information.
- FuSim, [Shen and Leitch, 1993], is a more recent qualitative simulator package with fuzzy capabilities. Fuzzy sets have been used in the definition of the finite number of values of a qualitative variable. FuSim allows to order the evolution of the states and transitions between them. This benefits the reduction of qualitative ambiguity and simulation more suitable for applications.

The main goals, in QR, have been centred on the domains of modelling, simulation and interpretation of systems behaviour using symbolic representation of variables. Pure qualitative approaches have two basic problems to be used directly in a model-based architecture for fault detection. First, ambiguity in the description of magnitudes and second the lag of temporal information. This causes, that mixed simulations (using numerical and qualitative methods) offer better results. Fuzzy approaches are also taken into account to smooth sudden transitions in the management of imprecision.

Despite of these drawbacks the idea of qualitative modelling can be used to establish a simpler relationship, not complete models, between qualitative representation of numerical process variables. The idea is to use this relationship on line with process as a qualitative observer, where a numerical relationship can not be applied.

3.4. Artificial Intelligence in Control Systems.

Since its beginning, AI has been present in control and process engineering. At that time AI was focused on learning architectures using connective systems called *artificial neural networks*. Some structures such as, *Perceptron*, by Rosenblatt in 1962, and *ADALINE*, by Widrow in 1962, are two representative examples of these epoch. From then until now, several approaches have succeeded in the AI domain to improve control systems according to two perspectives ; on one hand the techniques that try to mimic the expertise of humans and on the other hand the techniques based on machine learning [Årzen, 1995b]. In spite of the fact that this thesis is centred on the first approach (using expert knowledge to improve control actions and supervise process behaviour), a brief description of these approaches is given :

- *Learning control*, refers to control systems that are able to estimate unknown information during their operation and determine an optimal action from the estimated information. It is conceptually similar to classical adaptive control. Some learning algorithms are based on pattern classification, Bayesian estimation and stochastic approximation.
- *Neural Control*. This terminology refers to the use of artificial neural networks for learning process behaviours by training them. It has had an increasing popularity since the eighties. Various learning strategies have been developed for learning input/output relationship (*supervised learning*) or only sets of input (*unsupervised learning*) or delayed output (*reinforcement learning*) data. Multiple strategies can be found in the literature.
- *Fuzzy control*. This is the use of *fuzzy logic* for implementing direct control strategies. The idea is to describe control actions by means of simple rules applied to the input variables to decide about control action. It involves three steps : fuzzyfication, and defuzzyfication. Fuzzy logic is maybe the most extended technology, from the AI domain, in the industry. Nowadays, fuzzy logic is used in control systems where classical PID does not offer satisfactory solution (for instance, in non-linear or time dependent systems) as industrial controller or fuzzy logic unit into programmable logic computer (PLC).
- *Expert control*. The notion of expert control was originally introduced in [Åström, Anton and Årzen, 1986]. The aim is to use AI-base systems to supervise numeric control algorithms. These are systems designed to deal with both signal and symbols. The idea is to separate signal processing algorithms and logic. The implementation of logic is typically by means of rule-based ES. Some applications in this domain are auto-tuning controllers or fault diagnosis systems and

therefore, fall within the scope of this thesis. Example of an expert control architecture is depicted in Fig. 3.5.

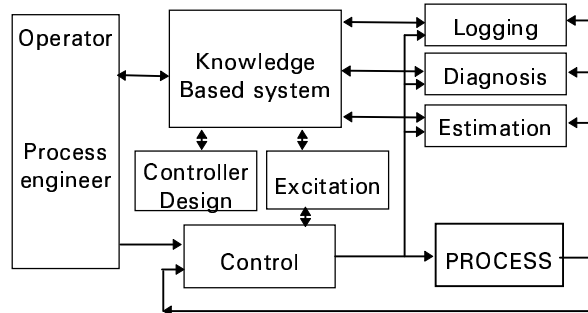


Fig. 3.5 Expert controller structure proposed in [Årzen, 1995b]

All of these AI technologies have been tested and applied with success in direct control applications. The difference between applying such technologies in a control strategy or in a supervision structure, is basically the kind of data to be used. Control is performed close to process and, therefore, input and output of controllers are numerical sampled data coming from, usually, one source. Simple numerical features (derivatives, filtering) are obtained from these signals before being processed by the controller. In supervisory systems more significant information must be obtained to be useful for automatic use by AI tools, because they try to identify those signals with structural changes, faults or localised missfunctions. Moreover, qualitative representations of these signals must be managed together with numerical values.

Expert systems were originally developed to solve static problems (situations where the premises do not change with time). Consequently, the introduction of ES in the control domain had some difficulties because of the importance of temporal dependencies of dynamic systems where control strategies are applied. Then, normally they are used in a higher level where time dependency is less than in direct control. Expert control falls within this direction and represents an example of how ES can be used for supervision.

3.5. Artificial Intelligence in Process Monitoring and Supervision.

Artificial Intelligence is introduced in process supervision because of its capabilities in dealing with different kinds of information [Gentil, 1996]:

- Heterogeneous kind of data (numeric, binary, qualitative, fuzzy). Significant information associated to process variables can be added.

- Imprecision and uncertainty.
- Incomplete sets of process data.
- Heuristics and relationship from human knowledge and experience.

The techniques used in this domain are extensive. Some of them have been tested in control application with successful results and they have been introduced in the supervision domain because of their capabilities in knowledge representation. In fact, the most used AI technologies in the domain of supervision, are real time ES. They have an extended use in fault detection and diagnosis. Others are only indirectly used. For example :

- *Fuzzy logic* is present in fault diagnosis in some applications for residual evaluation or threshold adaptation in the case of uncertain systems with changing operation conditions. Fuzzy logic is also used in the inference engine of some ES to deal with uncertainty and imprecision in data and rules.
- *Neural networks* have been used in supervision for residual generation and evaluation, in the fault diagnosis step. When models are difficult to be obtained, the use of neural networks can supply this limitation by training the network in normal operating conditions and using it as in a model-based approach. The other option is use the neural network to match specific fault situations. In this case, the neural network must be trained with data from previous faults. Thus, it is an alternative to analytical models in model-based methods.
- *Qualitative modelling* is present for obtaining qualitative models for simulation and variables estimation as a qualitative observer, see for instance [Melendez et al. 1996a] , where analytical methods are not available or numerical estimations are difficult to be applied. From previous presented simulation packages, integration problems are present because they are conceived as simulation tools and numerical to qualitative interfaces are not provided to be used directly connected with real data sources.

At the same time, the use of these techniques involve knowledge representation techniques, implicit in the application chosen for the implementation.

3.5.1. Integration problem.

Despite the great number of tools from the AI domain that can be used to represent expert knowledge and to make the procedure of writing rules and describing dependencies between variables, situations and causalities easier, supervisory systems design is still a research field in need of a methodology and frameworks where the complete cycle could be developed. Nowadays, design,

test, validation and implementation of supervisory strategies is not possible in a simple way using only one framework with necessary tools. Some commercial packages, such as G2, are as described before but still lack the basic tools, such as data abstraction tools for obtaining significant information (numeric and qualitative) or imprecision management in the rules description. Rule bases describing conditions related to both, numerical and qualitative description of process variables, is a common situation. In such cases, reasoning becomes easier if all possible information about these variables is encapsulated.

When additional capabilities are required, such as qualitative simulation or estimation, other packages must be used and data must be converted to be interpreted. The interface of simulation package and ES is not an easy task, and of course implementation of such systems becomes impossible. Moreover, if only simple relationships between qualitative descriptions of variables are needed, then the use of external complex packages is not justified and makes the option unrealisable.

3.5.2. Imprecision in temporal references.

The advantages of AI managing imprecise data become a serious drawback when translating this imprecision into temporal references. A message alarm about a possible fault “in 2 minutes” is slightly different from the message “is coming soon”. Similar problems are derived from the use of qualitative labels for representing process variables state during a period of time. Then, time dependencies are another important factor to be taken into account when dealing with dynamic process. Its importance increases when dealing with states transitions. A change in the behaviour from “normal” to “degraded” can not be suddenly. This kind of transitions must be smoothed in real-time operating conditions in order to detect these situations before they become irreparable.

3.5.3. Benefits of using OOP in supervisory tasks.

The main benefits of using OOP in the implementation of supervisory systems are derived from their characteristics, inheritance, abstraction and encapsulation of data and methods in the same structure. This applies in the construction of numeric to qualitative interfaces where numeric and qualitative data can be encapsulated together with the interface methods for an easier use. If additional methods for data access are provided to other objects, data can be shared and easily accessed.

Tools integration can be solved in this way if any data (numeric, qualitative, symbolic or logic) related to a process variable is encapsulated into objects and

all tools are provided with access methods for obtaining desired information related to these variables. This introduces the concept of *object-variables* explained in chapter five. The use of *object-variables* with their graphical representation simplifies the conceptual use of them. QR tools can be “connected” to them to access desired information for performing some action or operation. Other numerical tools connected to the same *object-variable* could operate with numerical attributes.

3.6. Conclusions.

The complexity of designing supervisory systems comes from the necessity of dealing with incomplete, imprecise and uncertain data and information related to process behaviour. Expert knowledge, from operators and engineers, must be introduced into computers, despite of the imprecision in the description of data and rules. Knowledge representation tools (logic, production rules, graph, frames and objects) are the interfaces used by AI tools with this purpose. Several AI techniques can help in this task if they are available in the same framework where numerical data is collected from process (measures, simulations or estimations) and specialised tools are provided to interface both numerical data and qualitative methods. The use of QR methods can assist supervisory systems design in the tasks of qualitative estimation (qualitative observer) of variables, more than for defining pure qualitative models for simulation or to be used in model-based approaches.

Numerical to qualitative interfaces are subjected to some drawbacks. The knowledge representation related to process dynamics needs the conversion of numerical data into qualitative one. Moreover, multiple qualitative representation can be necessary and coexist with numerical indices obtained from these signals. In such cases, encapsulation of methods and data is needed. Such interfaces must be very close to the process. Its inputs are numerical row data from the process and its outputs are thought to be connected to numerical processing and reasoning tools. OOP offers an adequate solution to this problem.

4.

CASSD - Computer Aided Supervisory Systems Design Frameworks

4.1. Necessity of CASSD frameworks.

Nowadays, many control applications take benefit of AI techniques for improving control algorithms or for developing expert knowledge-based controllers [Åström et al., 1993] [Lee et al. , 1993].

In the same line, AI techniques are often used in fault detection, diagnosis or supervision applications combining quantitative and qualitative information. Then, it seems to be useful to have a framework where several AI tools (for

instance, ES and QR tools) could be used friendly together to assist design and implementation of structures for special purposes and particularly, for assisting supervisory systems design. Consequently, a framework is required to achieve the kind of facilities needed for supervisory systems development. The aim is to have a framework where control and supervisory systems could be developed without the necessity of using external applications. Frameworks with these capabilities have not been found in the bibliography, although there exist some applications that link Computer Aided Control Systems Design (CACSD) environments with external KB or ES for reasoning. [James, 1988] provides a survey of coupled (numeric and qualitative) systems for CACSD. The idea of construction of most of these systems consist in linking external KB with existing packages instead of embedding KBs in the same framework. Consequently, these frameworks present some drawbacks:

- Engineers must learn to use more than one package to satisfy their needs, since no uniform, integrated design framework is provided.
- Transfer of data between different, often incompatible packages, is necessary; this slows down the operation, makes design more difficult and is a potential source of errors.
- The type, format and structure of data from different packages are often incompatible, and therefore, difficult to combine. Then, some drawbacks appear when comparing performances of different designs.

In most of these cases, interface and data type mismatches were solved by writing and reading files for data format conversions. A better solution would be to avoid data conversion between tools by providing a more efficient mechanism for data management.

4.2. Antecedents.

4.2.1. Steps towards a tools-based architecture.

In the late fifties and late sixties, computers used by control systems designers were difficult to interact. Due to their low processing power, they were used only to automate difficult numerical calculations. Meanwhile, controller design was done manually. In the late sixties, large computers allowed to perform large calculation by means of launching batch-operations. Control designers used those facilities for computing system responses in the time and frequency domains, root locus and simulation without the possibility of sharing applications and data. Those programmes, were developed to run only on a specific hardware and software platform. With the introduction of terminal access to main frame computers, during the mid-seventies, several numerical routines, essential in

FORTTRAN, were developed for solving control problems. Thus, development of computer-based control applications were converted to writing programmes where these routines were called. Difficulties occurred when experimenting with different algorithms (from different libraries) or parameters. Frequently, they resulted in source code changes and recompilations.

During the mid-seventies, research efforts in CACE (Computer Aided Control Engineering) focused on the interaction between control designer and CACE package. The resulting software was in the form of command-driven packages. The interaction was similar to that in interpreted languages, such as BASIC. Two commercial programmes built under command-driven approach succeeded in this epoch: CLADP (The Cambridge Linear Analysis and Design Programs) and a set of packages developed at the Lund Institute of Technology for modelling, analysis and design of control systems (IDPAC, MODPAC, SIMNON, SYN PAC and POLPAC). Both CLADP and the packages developed at Lund, used complex and arbitrary data structures which were not generally transportable. Users could define their own macros but the facility was not sufficient to build toolboxes. From the user interface point of view, maybe the main feature of the set of packages developed at Lund was the fact that it provided an integration mechanism (INTRAC) shared among several packages. The integration facility of INTRAC was only at the user interface level. This first intent in packages integration succeeded because all packages were developed at the same institute. Problems become bigger when integrating external packages from different vendors for getting a more powerful environment for assisting in all control engineer activities (identification, modelling, simulation, analysis, design, implementation, tuning, validation and documentation). Then, integration problems appear basically in two main points :

- Incompatibility of the user interfaces of different applications. Consequently, the user must know how to operate all of them.
- Incompatibility at the level of operating system. Data management and data format are not transportable.

The first solution to these problems was to provide data exchange facilities based on data translators (“Fig. 4.2a”) but then arose the problem of an increased number of packages to be integrate in the same environment. Consequently, attempts on integrated environments focused on this topic and some projects appeared, incorporating enhanced database management functionality for supporting data relations and project design histories, while taking maximal advantage of existing software modules. The earliest integrated environment of this type was the Federated Computer Aided Control Design System described by Spang in 1984. This has been followed more recently by an Environment for Control System Theory And SYnthesis (ECSTASY), [Munro, 1990] and the Multi-disciplinary Expert-aided Analysis and Design (MEAD)

environment proposed, by Taylor and McKeehen in 1989 (This is clearly described in [Taylor et al., 1990]). The basic idea of the architecture of these systems consists in the integration of external packages as is depicted in Fig. 4.1 b). The user communicates with the packages through a command language interpreted by a supervisor that performs translations to package commands. A similar procedure solved the transfer of data from packages to a common data base to be shared with other packages. The main disadvantage of this architecture became obvious when new enhanced versions of the constituent packages appeared with better functionality and capabilities. The integration framework could not follow continuously these changes and the maintenance of such environment was impossible.

The next step in the evolution of CACE frameworks was firstly proposed by Anderson in 1991 and further refined by Barker in 1992. The proposed architecture (“Fig. 4.1c”) consisted in integrating not complete packages but simple tools (Tools-based architecture). The environment provided user interface and data storage and management services. A more extended description of these proposals is included in [Jobling et al. 1994].

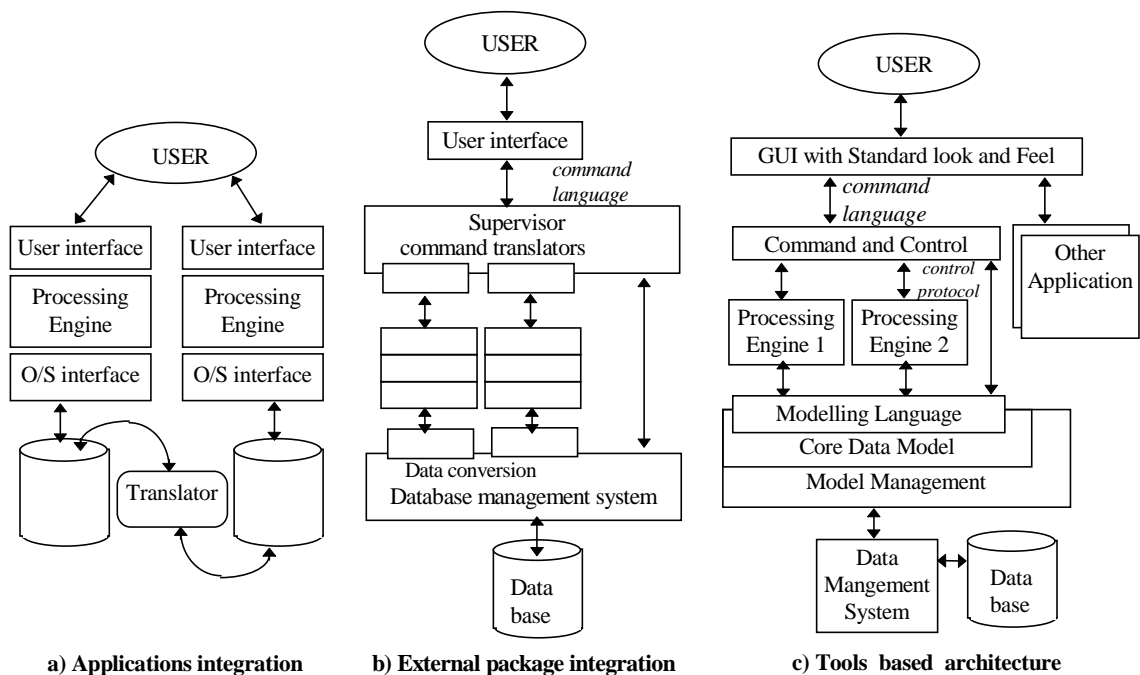


Fig. 4.1 Solutions for packages integration.

The tools-based approach has been taken as reference for posterior works taking advantage of object-oriented programming for designing these tools. Thus, in an object oriented architecture tools are objects containing models as

attributes and these objects respond to user messages sent by the user operating the user interface. In [Barker, 1995] basic objects defined by dynamic systems models are proposed for representing plants, actuators, controllers, sensors and any abstraction of physical elements to be used in the design.

As control implementations grew in size and complexity, CACE tools assumed a more important role in engineering design and implementation of computer controlled devices [James et al., 1995]. Consequently, in the middle eighties an increasing number of applications and environments appeared in the bibliography. Special interest, in the majority of these works, is centred on assisting the design stage of control systems. These are the Computer Aided Control Systems Design (CACSD) environments. Some of them are especially oriented to process, for example [Ogunnaike, 1995] presents an environment especially conceived to improve the control of a commercial polymerisation reactor. Meanwhile, other environments are focused on specific control methodology, such as is the software presented in [Bohn and Atherton, 1995] to compare PID anti-windup strategies.

In fact, topics related to CACSD and CACE environments domain are in constant evolution to adapt contemporary software technologies and to satisfy control engineers requirements. Object-oriented programming is one of the topics that has hardly been introduced in new developments as integration methodology. Authors as Jobling believe that an object-oriented view of CACSD is necessary to reach the complete functionality of these frameworks [Jobling et al. 1994].

4.2.2. The reference model for CACE open environments.

The foundation of all forms of CAE (computer aided engineering) is the environment, which contains the software to support the engineering of a product or process throughout the its whole life, from conception and specification, through design and development, to implementation and operation. In the case of control engineering, the final product is the implementation of control algorithms designed to reach a specific goal. Thus, the basis of a way forward in CACE open environments could be inherited from the experience in CASE (Computer Aided Software Engineering) environments, since they are the most technically advanced software development products and receive the greatest amount of investment.

Nowadays, it is widely accepted that it is necessary to provide CASE environments with a set of facilities, or services, (known as a Public Tool Interface) for assisting tools management. This is a set of services joined in a framework that, when adding the appropriate tools, constitutes an environment.

The European Computer Manufacturers' Association (ECMA) adopted in 1990 a reference model as standard Public Tool Interface for CASE environments. In 1991 the American National Institute for Standards and Technology (NIST) also adopted the same reference model [Barker, 1995].

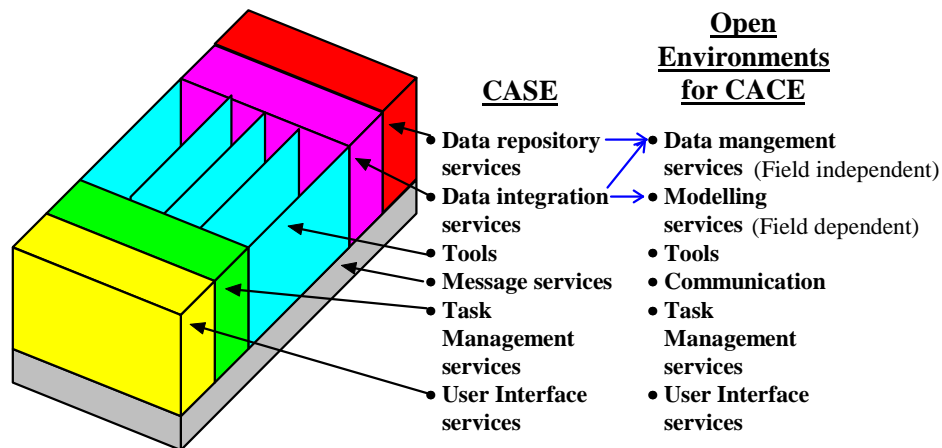


Fig. 4.2 Comparison between the Framework Reference Model for CASE and Open environments for CACE.

The common facilities proposed by this reference model were divided into five groups of services (See “Fig. 4.2”).

1. *Data repository services*, which provide storage for all data objects in the environment.
2. *Data integration services*, which enhance the services of the data repository by inserting a layer of abstraction to provide high level semantics and operations for handling the stored data.
3. *Message services*, which provide managed communication between all the facilities in the environment.
4. *Task management services*, which create a task-oriented environment by inserting a layer of abstraction between the user and the fine detail of the facilities in the environment.
5. *User-interface services*, which provide a consistent graphical user interface for the whole environment.

The open systems approach for CACE integrates those data services that are not specifically field dependent in a field-independent group of database services. The remaining data integration services form a field dependent group of modelling services for modelling the physical world. The benefits of open systems is clear since they enable any tool from any vendor to work cooperatively with other tools by sharing common data formats, control mechanisms and user interface (See “Fig. 4.2”). From this point of view ANDECS ([Grübel, 1993] and [Grübel, 1995]) could be taken as a representative

example of open CACE environments architecture. ANDECS provides the following seven classes of services:

1. Database services
2. Model-definition services
3. Algorithmic services
4. Tool-control services
5. Task-control services
6. User-interaction services
7. Process-communication services.

The functionality of these services is directly deduced from the previous description. They provide the necessary infrastructure for “bottom-up” evolutionary software engineering for CACE applications: They support flexible-to-use tool systems where the application functionality is embodied in the tools. These tools are thought to operate stand-alone or within tasks of computation chains and loops. This is because ANDECS is actually taken as the reference point when talking about open CACE environments.

4.2.3. CACE and CACSD packages.

In the bibliography sometimes the difference between CACE and CACSD is not clearly specified and the two groups of systems are confused. As a consequence, they are used indistinctly for denoting an environment especially conceived for assisting control engineers. But, in fact, the strict terminology assigns different roles to these words: while CACSD packages are used to design collections of tools for computational execution of control and system theory, CACE frameworks allow the integration of such CACSD tools with more general modelling, simulation and optimisation tools [Grübel, 1995]. This is the way in which CACSD and CACE have been used in this text. According to these differences, the majority of environments presented in the bibliography for assisting control design in a specific domain are CACSD environments, since they are conceived as a set of callable tools (routines) that do not match the reference model presented in the last paragraph.

From the engineers' point of view, CACSD are expected to provide the necessary tools for associating process elements with simple representations and some facilities for building and manipulating these models. With this scope, basic and general features for CACSD environments were defined in [Jobling et al. 1994]:

- sophisticated graphics to ensure that the model "looks" and "feels" right;

- excellent data handling which serves as the necessary basis of the instrumentation of the model;
- powerful manipulative software so that the models can be simplified for analysis and design
- software support for the design process itself in order to correctly identify versions of the models with the results generated and design decisions made.
- powerful modelling paradigms in order to ensure that physical behaviour of the system can be correctly identified and represented.

A relevant set of CACSD systems is presented in a special issue of the Control Systems Magazine [IEEE Control Systems, 1995]. Some of them are open environments that offer the possibility of linking with other external packages, but they are not built according to the reference model of CACE environments (for example, in the mechatronic domain the package CAMEL (Computer Aided Mechatronic Laboratory) [Rutz R. and Richert, 1995] supports all the design cycle of a mechatronic system in an open CACSD system with access to external tools as MATLAB or MAPLE). Others are conceived as sets of routines added to a commercial framework for assisting special control systems design and analysis (for example, a special Toolbox is added to MATLAB/Simulink for analysis and comparing PID anti-Windup strategies [Bohn and Atherton, 1995]).

4.2.4. MATLAB/Simulink as a CACSD framework.

This section is especially devoted to discuss about one of the most popular frameworks in the control community, MATLAB(Matrix Laboratory)/Simulink, and its importance as a CACSD framework. In fact, a recent questionnaire of the Working Group on Software (WGS), created by The Commission of the European Communities to co-ordinate the development of robust numerical software for control systems analysis and design, has noted that MATLAB is the most extended package in the control community (See [WGS Newsletter, 1997]). MATLAB is a command-driven open system especially conceived for matrix manipulation and visualisation, and its extended use is due to its characteristics as [Jobling et al. 1994] pointed out below:

- It supports simple and yet fast and flexible command language interface;
- The command language is interactively extendible ; and
- the basic data structure (the complex matrix) corresponds to the primary data structure used in modern state-space control theory.

The MATLAB syntax is a de facto standard in matrix-based linear systems and control theory. Therefore, a CACE framework must assist the use of this syntax [Grübel, 1995]. Thus, for dealing with linear systems and classical control theory,

MATLAB seems to be a reference point. In fact, nowadays, it is unquestionable that MATLAB can be defined as the state-of-the-art in CACSD, [Jobling et al. 1994]. The increasing number of ToolBoxes (a Toolbox is a set of routines, directly callable from the MATLAB command line, focused on a special topic) related to control domain that have appeared in the last years strengthen this affirmation (for example [Bohn and Atherton, 1995] and [Chipperfield and Fleming, 1995]). From CACE point of view, MATLAB is always taken into account as a tool to add in the framework. In fact, frameworks as ANDECS, CAMEL and GE-MEAD offer services to link with MATLAB.

In spite of this extended use of MATLAB, certain drawbacks of the system became obvious when dealing with some complex systems. This is resulting from the poor support that MATLAB provides for data structures, such as transfer functions, non-linear system models and signals, which can not be manipulated as easy as matrices. MATLAB is frequently criticised for the poor support for representing complex structured entities and type checking [Saifuddin, 1996] needed when dealing with large control problems, where control entities should be represented by data structures to simplify. This has been partially solved within the new version of MATLAB v. 5.0 that supports more complex data-structures (not really objects, because encapsulation of methods and data is not supported).

From the CACSD point of view, a friendly user interface is needed. The command-line interface provided by packages like MATLAB, is enough for simple applications and the possibility of group commands in script files enlarges the use of this framework for more complex applications. However, there is no doubt that the interactive graphical user interface offered by Simulink (extension of MATLAB that provides a block representation of many functions defined in MATLAB) allows users to input block-diagram models by simply dragging the mouse to interconnect blocks. Simulink is not the only user interface with this capabilities. Other similar graphical user interfaces include SYSTEM -BUILD (ISI 1989) for MATRIXx and Model-C (SCT-1987) for CTRL-C.

Despite that MATLAB/Simulink has some difficulties in providing clean functions for dealing with the more complex data objects that are found in control systems analysis [Jobling et al. 1994], it has been chosen as a platform to develop a Computer Aided Supervisory System Design (CASSD) framework [Melendez et al., 1996b]. With this approach engineers can proceed first, to design a control system, and second, to design a straightforward supervisory system of the control system in the same framework. Simulink, instead of MATLAB, has been chosen for this purpose because it provides an intuitive, user friendly graphical environment employing visual representation widely accepted in the automatic control domain. The use of block representation helps engineers

to order ideas (it forces to analyse before they start connecting blocks) following a declarative methodology. Thus, if a set of tools is available as simple blocks and its use is as simple as selecting them and interconnecting in causal order, then design becomes easier. The next chapter explains how some tools, from the AI domain, have been developed as Simulink blocks with the special purpose of assisting engineers in knowledge-based designs. The previous described disadvantages have been overcome by using the object oriented approach in the definition of variables managed by Simulink, which is also explained in the next chapter.

4.3. Coupled CACSD frameworks.

The majority of CACSD frameworks have been developed to assist control systems design from the point of view of classical control theory. In these cases, only numeric data is managed and simple data structures, such as matrices, were sufficient to perform a flexible mechanism for data management and model representation (mainly transfer function or state space representations are used.). As a consequence, popularity of some frameworks, such as MATLAB or MATRIXx has increased and they are evolved to complete environments with modelling and simulation tools and advanced mathematical analysis and visualising tools supported by a block based graphical interface. The main shortcoming of these frameworks consists in poor support for use of advanced contemporary control techniques. This applies especially to the tools inherited from the AI domain. The reason of this lack is basically the heterogeneous types of tools and, consequently, the different types of data used by these tools. In fact, the only modern techniques that have been introduced in these frameworks are those that support a matrix representation as neural networks or fuzzy logic. Consequently, problems that have not a description by means of classical formalism cannot be solved in these frameworks. Hence, it is clear that the benefits of adding new approaches to these frameworks may be significant; the new, extended systems would allow for combining symbolic methods from AI research with more traditional numeric methods obtained from numerical analysis, operation research and simulation. The result appears to constitute a more powerful and more useful problem solving the environment [Jacobstein and Kitzmiller, 1988].

Historically, two reasons have been pointed to merge symbolic and numerical approaches: the first, when domain-specific knowledge is needed and the second, to guide non-expert users in problem-solving steps to fit the best solution. Complete environments have been developed to solve specific problems, coupling ES and numerical algorithms. Other attempts in coupling numerical and symbolic information have been done linking independent packages. A survey of

coupled systems for CACSD is provided in [James, 1988][James et al., 1995]. In all the proposals studied in that paper, a knowledge-based system is added to aid engineers in the design of control systems. The majority of those systems have implemented or intended to implement some method of integrating the symbolic processing of expert system shells with numerical processing of conventional control design software. Table 4-1 summarises the projects overviewed in that paper.

Program Name	Inference engine language	Knowledge representation technique(s)	Numerical analysis routines	Coupling method	Intended program Use
CACE-III	VAXLISP	Rules	CLADP SIMNON SSDP SFPACK	files	design assistant or intelligent interface
BOFFIN	C, LISP	rules, frames and blackboard	custom	function calls	guidance and control for autonomous submersibles
IHS	FRANZLISP FLAVORS YAPS, C	Rules and Scripts	IDPAC	VAX/VMS	demonstration
CASCADE	PROLOG	decision expressions	CTRL-C	link between processors	demonstration, adaptive control
Lockheed Flight Controls Workstation			EISPAC LINPAC ORACLS IMSL MATLAB		flight control systems design
MEAD	VAXLISP	rules	CTRL-C ACSL SIMNON MATRIXx		flight and engine controls
Control Design Expert System	LISP	rules and frames	custom	function calls	demonstration only
ROBEX	C	rules and/or trees	custom	function calls	design assistant or tutor
CORTEX	PASCAL	rules	custom	function calls	self-tuning control
SFGI	LISP	rules	TURBO PASCAL	DOS-Call	design assistant

Table 4-1 Coupled CACSD programs. Extracted from [James, 1988].

Knowledge representation methods used in those projects include rules, frames, and/or trees, predicate logic, scripts and objects. While the majority of available shells offer a single knowledge representation scheme, some attempts have been made to offer a combination of knowledge representation methods.

The basic advantage offered by these schemes is that they allow the programmer to use the method which best fits the problem at hand. On the other hand, the majority of these projects have been developed using existing numerical calculation packages that were conceived to be used as a single application. In consequence, coupling different methods obstruct the use of the whole system since at the same time several packages must be learned by the users.

James pointed one reason more for applying the knowledge-based approach to CACSD programming [James, 1988]: The ability of knowledge-based systems to deal with complexity. The AI techniques offer the considerable advantage of being easier to use in complex applications. Thus, it seems correct to suppose that in developing supervisory systems, the availability of a specific environment to assist those tasks would be extremely advantageous. Moreover, it would be desirable that this framework would be provided with a graphical user interface and all facilities embedded in the same framework to facilitate its learning and use.

4.3.1. Knowledge-based systems and MATLAB/Simulink.

If MATLAB is taken as the state-of-the art in CACSD, as is described in paragraph 4.2.4, then it seems logical that some intents of adding AI features to this framework would improve performance when working with complex systems. In fact, some examples of linking MATLAB or Simulink with an external KB can be found in the bibliography. For instance, in 1987, Butz at Temple University in Philadelphia, linked the OPS-5 expert system shell, under VMS, with MATLAB and Ctrl-C to design a phase-lag compensator. MATLAB was used as a numerical analysis package and the ES took care of symbolic management of information. Both packages were interfaced by means of a file transfer system. Another system used MATLAB as part of an ES for designing lead-lag compensators based on given specifications. This is called CDES (Controller Design Expert System) and is described in [Ong, 1992]. In this case, the knowledge base is an external module and MATLAB is used only for numerical calculation of performance and representation. Subsequently, MATLAB/Simulink and the shell G2 were linked by means of a synchronous communication for developing a fault diagnosis system [Melendez et al. 1995]. Later, the same application was developed using the shell CEES (C++ Embedded ES, [De la Rosa J.LI, 1994]) in the same group at the University of Girona. In this case Simulink was used as simulator. Abstraction tools were developed under Simulink to obtain significant qualitative information from signals. This information was directly supplied to the ES during the simulation.

The described examples always focused on linking MATLAB or Simulink with external shells. MATLAB/Simulink was only used for numerical analysis. In

consequence, work with these environments requires always to know two or more different packages, and sometimes interface mechanisms are not transparent to the user. Nowadays, most of commercial shells are provided with mechanisms for exchanging data which reduces the interface problem to simple programmes for data conversion and client/server communication. But, the problem of using more than one packages still exists. Then, the proposal made in this thesis of embedding the ES into a CACSD framework (for extending the use of this CACSD to expert supervisory design) with graphical user interface, seems to benefit the users because it is still easier to use. For details about this topics see section 5.5. The selection of Simulink to test this idea is reasonable since it offers this graphical user interface and it is presented as an open system from the point of view of adding new functionality by means of using standard languages as FORTRAN, C or C++.

4.3.2. Characteristics of Expert Systems for CACSD.

Regardless of the attitude towards the role of knowledge-based system as design assistant, as an intelligent interface, as an autonomous or semi-autonomous adaptive controller or as a tutor, the expert system shell should provide the following general features ([James, 1988]):

- Since the design process is an iterative one, the shell should provide for the revision of previously believed facts (e.g. changes to specifications or changes to parameters values) as the design process proceeds.
- Since the design knowledge contained in an expert system can become very large, some method should be available to support the decomposition of the knowledge into manageable pieces.
- Given that many intermediate designs may be generated, the data structures of the expert system should support the comparison required for trade-offs.
- The explanation facility of the expert system should support a hierarchy of levels of explanations which corresponds to a hierarchy of progressively more fundamental justifications for steps in the design process.
- The expert system shell should be open in the sense that the user has the option of customising the shell to his own application.

These points must be taken in account, not only for selecting an ES shell, but to focus on the development of the whole knowledge-based CACSD. In fact, they are not restricted to control systems, design but in any application where an interactive procedure is performed before reaching the final design. Sometimes the use of external shells linked with a CACSD framework obstructs the conservation of these points for the whole environment, because of dealing with

several independent applications. Then, the best solution of merging knowledge-based capabilities with CACSD frameworks is to embed rather than to link the ES into the CACSD framework. Especially if ES capabilities are not essential for the design but are necessary as a tool to be incorporated in the design. Then ES must be embedded as another tool in the framework. Thus, the addition of ES in a CACSD with a graphical user interface must be performed preserving the same interface as much as possible. This is the solution adopted in this work. It was previously introduced in [Melendez et al., 1996b] and presented in the complete framework in [Melendez et al., 1997a]. The inference engine of the shell CEES has been embedded in Simulink as a new block because this offers an easy-to-use user interface and multiple block-based tools. Additional features and configuration options that do not have a block interpretation have been added using the menu bar. The advantage of using block representation is clear from the user point of view. Additional advantage of this representation dealing with ES is that large knowledge bases could be structured as the simplest ones. Fig. 4.3 represents this characteristic with an ES reasoning about block input facts to deduce output facts.

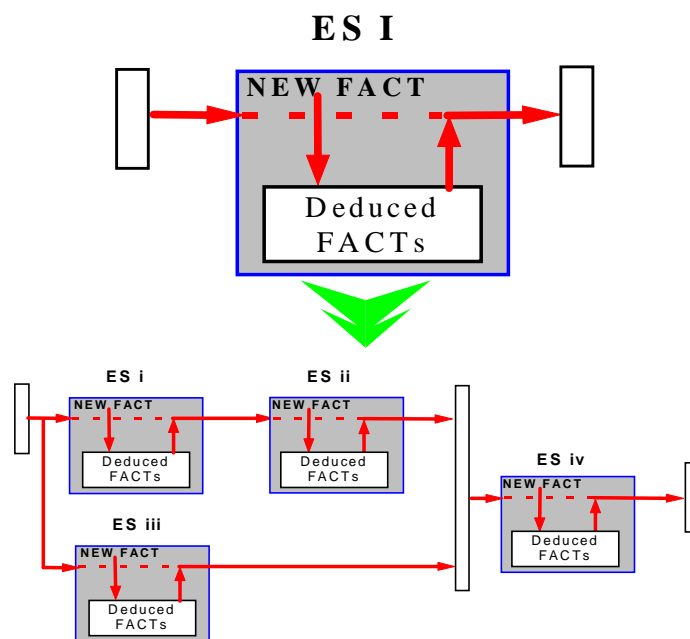


Fig. 4.3 Block representation allows engineers to structure knowledge bases.

The fact of using Simulink as support framework makes it easier to accomplish the previously pointed features, because some of them are already available. For example, the comparison between different solutions is performed simply by adding both designs into the same window and taking benefit of the representation capabilities of this framework.

4.4. Proposal for a Computer Aided Supervisory System Design (CASSD) framework.

Previous sections have shown the benefits of using specialised frameworks for solving engineering problems. In control domain the number of this kind of frameworks have increased in last years. They are especially centred on classical control theory. Although some attempts of adding AI capabilities have been done with different goals in the control field, there is not any specialised framework for assisting expert supervisory systems design. Two main reasons can be pointed :

1. The heterogeneous nature of the data involved in these designs. According to the kind of application to be developed it is necessary to start defining the kind of data to be used, declaring data types and assigning labels to symbolic variables for representing qualitative trends.
2. The majority of environments presented before are formed by linking external applications, which results in a interfacing problem that, at the end, must be solved for any new design.

Then, the proposal presented in this work tries to solve these two problems. At the same time, the experience accumulated in CACSD frameworks development is taken in account in this work.

4.4.1. Requirements for a CASSD framework.

Since expert supervisory design could be viewed as a way to improve control systems by means of AI capabilities, it seems to be succeeded in expanding CACSD frameworks by adding AI tools. Limitations in actual CACSD frameworks appear with the necessity of incorporating reasoning tools and knowledge representation. Consequently, basic facilities described in [Jobling et al. 1994] for CACSD environments are kept for CASSD environments and new requirements are added. The basic facilities inherited from CACSD frameworks are :

- sophisticated graphics to ensure that the model "looks" and "feels" right;
- excellent data handling which serves as the necessary basis of the instrumentation of the model;

- powerful manipulative software so that the models can be simplified for analysis and design
- software support for the design process itself in order to correctly identify versions of the models with the results generated and design decisions made.
- powerful modelling paradigms in order to ensure that physical behaviour of the system can be correctly identified and represented;

It is expected that a CASSD framework becomes an extension of CACSD with additional specific features for assisting supervisory structure design at any step. Capabilities for managing (modelling and simulation), processing (analysis, estimation, calculate, etc.) and representing (graphical visualisation tools) numerical data (present in the actual CACSD frameworks) are needed to implement classical analytical approaches, such as model or signal based fault detection. Despite analytical based fault detection theory has been widely developed, it can not be applied in the majority of cases, neither in fault detection nor in more complex supervisory structures. Difficulties of obtaining an acceptable model do not allow to use such techniques.

Therefore, AI tools must be added to this framework for dealing with knowledge representation and processing. Since CASSD is required for representing knowledge about process behaviour, these mechanisms must be focused on both, process variables and human knowledge. Process variables are supplied as numerical data while human descriptions are normally given as rules. Another important aspect to be taken in account is the difference between the developing and execution time of knowledge-based systems. Despite any knowledge representation can be used for structuring knowledge, actual AI techniques lead to ES based applications because of its knowledge processing capabilities.

Expert knowledge from the engineers is not always given as quantifiable magnitudes, but as case descriptions or representative activities. For assisting this knowledge representation, the CASSD framework must be provided with some mechanisms to force engineers to structure their knowledge in the design time. Formal knowledge representation methods (logical formalisms, production rules, semantic networks, frames or combinations of them) must be implemented in an easy-to-use form transparently to the user. On the other hand, information about process behaviour is extracted basically from process variables (measures, simulations or predictions). This is numerical information that must be interfaced with expert knowledge. In the execution time, this data will be processed to obtain significant information according to expert knowledge. The CASSD framework must provide these mechanisms to abstract information from row data (*abstractors* or *abstraction tools*) needed as interface between numerical process

variables and qualitative interpretation of process behaviour done by process engineers.

Because not all the expert knowledge related to process variables is accessible, a CASSD framework must be provided with some mechanism to deduce not accessible dynamics. Numerical simulation or prediction could be useful when a model is available. In other situations, some qualitative reasoning mechanism, qualitative modelling or qualitative observers, should be available for roughly deducing qualitative evolution of process variables (qualitative magnitudes or trends). In many supervisory applications, this rough information is enough to deduce a malfunction or to generate an alarm.

According to the described problems when dealing with this heterogeneous sources of information, many different tools are needed to be integrated in the same framework to facilitate:

- *Numeric/Qualitative interfaces.* Process variables are the main link between process and supervisory system. Then knowledge processing tools must be provided with qualitative representations of process variables. These interfaces must be designed to provide multiple representations and feature extraction from numerical variables. Not only qualitative representation is needed, but also numerical features as mean values, statistics, trends, regressions and so on. They can be instantaneous features, parameters related to history of signals or predictions and estimations of trends and numerical values.
- *Knowledge representation.* Several tools can be used to deal with expert knowledge at several abstraction degrees. Knowledge of process can be about process variables, the relationship between them, description of situations (faults) related to physical elements, functionality, behaviour and so on. All of these possibilities require some specific tool with different representation capabilities to describe trends of variables, relations between them or process states related to a set of process variables.
- *Knowledge processing.* Usefulness of knowledge-based systems resides in the capability of deduction using knowledge representation. Then, those inference engines must also be present in the framework to deal with qualitative relationship (*qualitative reasoning*), rules (ES), logic, and other possible knowledge representations.
- *Uncertainty and imprecision management,* is necessary in this framework because these are inherent to qualitative information. Then, knowledge representation and processing tools integrated in the CASSD framework must be provided with mechanisms to process this imprecision.

Some tools are selected and presented in chapter five to assist in the design of supervisory systems. The goal is to take advantage of the existent representation and analysis tools used in control systems design, extending its capabilities with knowledge-based techniques to assist engineers in such designs avoiding the always difficult problem of interfacing separated tools reasoning on dynamic systems. A possible framework, based on MATLAB/Simulink, with these capabilities has been presented in [Melendez et al., 1997a]. The main work has been centred on solving the principle drawbacks in knowledge representation and processing, and the integration of these into a numerical framework.

4.4.2. Selection of a platform.

For developing the previously presented idea of CASSD platform, two solutions are possible. One of them is to develop a completely new framework to assist expert supervisory systems design. The second possibility is to take advantage of an existing CACSD framework and try to implement the required facilities within it. The first solution can lead to implement a commercial package, but in this the work second option has been selected in order to achieve an open complement of the widespread use of MATLAB/Simulink environment in process control design. Moreover, the goal of this thesis is to present some ideas for integrating useful tools for supervisory systems design. Therefore MATLAB offers the possibility of adding and combining new approaches to improve the framework.

According to the features proposed in the previous paragraph for CASSD frameworks, basic facilities are inherited from CACSD packages. Therefore, a commercial CACSD package, MATLAB/Simulink, has been used as CACSD framework and knowledge representation capabilities and qualitative reasoning tools have been integrated by means of an object oriented approach [Melendez et al., 1996b] to solve integration problems. MATLAB has been chosen because of its openness, and facilities and because it is representative as a CACSD package, as described in section 4.2.4.

4.4.3. Other frameworks with similar capabilities.

Other packages that could be used as support in this thesis such as G2, from Gensym, or MATRIXx, are not so extended in the control community. In fact, Gensym offers an extension of G2, called GDA (G2 assistant for diagnosis), that improved G2 features with a graphical block representation and some advanced features that include fuzzy logic, neural network and temporal logic. With this toolbox, G2 offers a complete environment to build applications for smart alarming (filter events and alarms) and advanced control (statistical, neural, fuzzy

control). This is thought to be a shell for intelligent monitoring developments. It is especially conceived for dealing with numerical data, for sensors validating, trends management and event discrimination, and managing information flow to operators.

Limitations of this package with respect to the proposed approach are centred on the dealing with qualitative information. Without qualitative reasoning capabilities, the qualitative representation of information is restricted to text labels description. Although GDA incorporates fuzzy logic to deal with imprecision in numerical data, the inference engine does not incorporate this capability to deal with certainty in the rules description and data management. Despite of these drawbacks, G2 could be used as starting point but was in the first place excluded because of its poor facilities for representing, analysing and processing numerical data. An additional reason to discard this framework as platform to implement the ideas developed in this thesis, was because it is more process-oriented tool than to control, and the original ideas of this work were developed to deal with expert supervision of controllers.

Another package to be used in this purpose is MATRIXx. It is an open system with similar capabilities to MATLAB/Simulink and more control oriented than G2. The main drawback resides in that it is less extended in the control community and maybe it is more a device oriented package.

4.5. Conclusions.

The necessity of a framework for dealing with expert knowledge when designing expert supervisory systems has been introduced in this chapter. This necessity is clearly derived from difficulties in knowledge representation and interfacing. The development of expert systems and knowledge-based application for reasoning about dynamic systems do always requires numerical to qualitative interfaces.

From a historical point of view, computer aided design has been introduced in all specific domains where the use of adequate tools can help in the configuration and building of specific applications. This is the case of control systems design (CACSD) and other disciplines where engineers are involved.

Basic CASSD requirements for dealing with expert knowledge in control systems have been inherited from previous CACSD experiences, adding AI features for knowledge representation and reasoning and to facilitate tools integration avoiding data conversions. Graphical capabilities are expected to be

present in the management of such tools with the aim to facilitate engineers in structuring and representing acquired knowledge. Modularity is also thought to be present for better knowledge representation and partial validation.

5.

Integration of tools for supporting supervisory systems design

5.1. Introduction.

A set of tools from the AI domain has been integrated to sustain supervisory systems design. Both numerical and knowledge-based tools are thought to be useful for this purpose. Therefore, a qualitative representation language, called ALCMEN, and a shell for developing ES have been added to an existent CACSD framework. The goal is to take advantage of graphical representation mechanisms and numerical methods used in the design, analysis, test and validation of control systems. The use of MATLAB/Simulink with this purpose offers the possibility

of a block based graphical user interface. Some interfacing problems appear due to different capabilities in data management, and representation, of numerical and knowledge-based tools. Difficulties in data integration have been the main reason why CAD has not been extended to supervisory systems domain. With the aim of offering a simple approach to solve this question, *object-variables* have been proposed in this work, and presented in this chapter, as objects encapsulating multiple representations of process variables. Different representations are possible using adequate algorithms for isolating desired features of signals. These algorithms are called *abstraction tools* and in this chapter are proposed also as numeric to qualitative interfaces.

5.2. Sharing Information between tools : *Object-variables*.

Data encapsulation is one of the basic features of OOP. That is why the classical approach of OOP in large applications ensures better structuring and organisation. In this work, encapsulation is used to offer a multiple representation of data for avoiding data type conversion and additional processing using object structures in the definition of process variables. With this goal the concept of *object-variables* is introduced. The aim of *object-variables* is to provide the adequate representation of process variables for any tool that require information related to them. An additional contribution of this work has been done in demonstrating the viability of using these *object-variables* in a commercial package, MATLAB/Simulink working together with numerical and knowledge-based tools under a graphical block representation.

5.2.1. Using process variables for reasoning.

Expert reasoning about process behaviour is usually not only associated to selected, isolated values of process variables at a certain instant of time, but also takes into account the whole process dynamics, including their history, trends, numerical and qualitative characteristics, limitations, etc., together with their influence on the behaviour of other individual process variables. Such inference, therefore, becomes easier if all information from the process is clearly encapsulated and located [Lynch and De Paso, 1992] [Jobling et al. 1994]. Organisation of process related information must be done according to multiple sources of information (sensors, analytical models, human experiences, etc.), then different units for encapsulation of information can be defined. Taking this into account, it seems reasonably that all information concerning process variables should be encapsulated in template structure. Thus, any information related to this process variable would be accessible. This means that the values of process variables supplied by sensors or simulation tools should be encapsulated with all

information related to them. Parameters, units, landmarks, qualitative characteristics and additional knowledge supplied with the variables can facilitate the interpretation of process behaviour providing all the auxiliary information concerning the analysed signal grouped together. Further, when designing architecture for supervision and control, sometimes different tools can use the same variable values, but simultaneously the variables are analysed at different level of abstraction (depending on current needs, numeric values, tendencies, deviations, qualitative values, event generation, alarms, etc. can be considered.). See Fig. 5.1. Unfortunately, those different tools are not provided with methods to obtain this information from purely numerical values given by sensors. This auxiliary calculation must be performed outside of those reasoning tools what further causes an interfacing problem to occur.

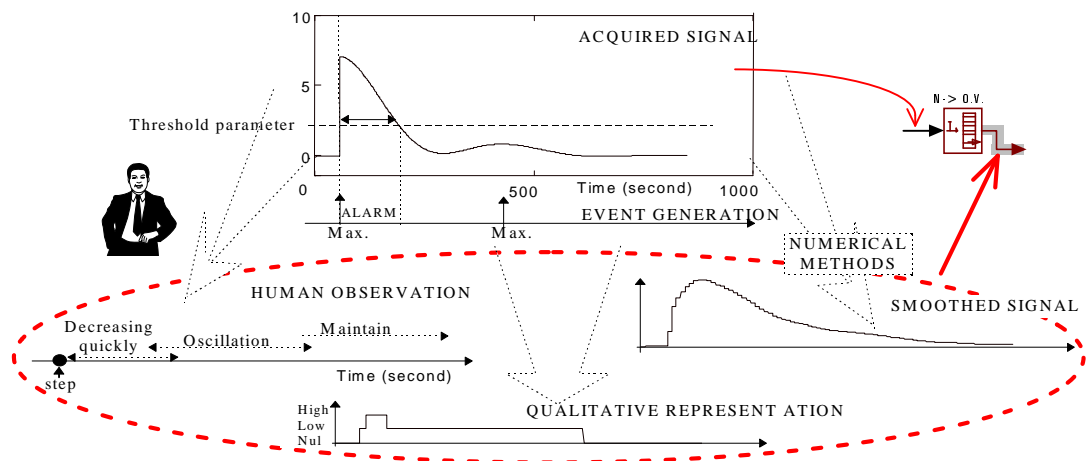


Fig. 5.1 Multiple representation of a process variable and information related.

To summarise, the problem consists in the necessity of multi-aspect representation of knowledge about process variables. The main issues include the following:

- *Encapsulation of knowledge* : all knowledge referring to a specific variable should be contained and accessible from/through the access to this variable only, so that the search of many related sources is eliminated. For example, the threshold used in alarm generation in Fig. 5.1.
- *Multi-level knowledge representation* : knowledge about process variables should be represented at several levels of abstraction, so as to enable numerical, qualitative and symbolic information processing when necessary, without drastic changes in knowledge sources and knowledge representation mechanisms in use (Evolution of signal in Fig. 5.1 has a smoothed and a qualitative representation associated to the numerical acquired signal)

- *Multi-aspect knowledge representation*: all necessary aspects of knowledge representation should be covered, so that variables can be analysed, taking into account the specific perspective and requirements of different tools applied for knowledge processing. For example, associated alarms and events related to process and acquired and smoothed signal in Fig. 5.1.

The object oriented approach seems to offer the solution thanks to its capability of encapsulating structured data and methods in the same structure. Using this approach in the definition of template structure for process variables, basic necessities, as mentioned above, can be covered.

5.2.2. Process variables at different levels of abstraction : *Object-variables.*

The proposed solution for solving the necessity of this multi-aspect representation of process variables presented in this thesis consists in embedding not only numerical values of variables (measures, estimations, history, indices and so on) and parameters (thresholds, landmarks, ranges, etc.), but also methods for obtaining the adequate description of signals behaviour for each tool (signal processing and reasoning algorithms, abstraction tools), and methods for manipulating and accessing this information.

Hence, variable values, parameters and methods will be encapsulated in the same object structure, to be called an *object-variable* [Melendez et al., 1996b]. See “Fig. 5.1 and Fig. 5.2”. Thus, the same variable is shared by various tools providing the adequate information (representation) for each of them. The Object-oriented approach gives the possibility of representing one variable as a single entity with its multiple representation and related information, such as methods and parameters, in a very flexible and powerful way.

[Årzen, 1995] uses a similar proposal for representing process components that can be used in a different context and that can only be represented once in a knowledge base (they are called *multi-view objects*). *Object-variables* are used with a similar goal to encapsulate in one object information and representation of the same variable at different levels of abstraction. Whereas *multi-view* objects are focused on the description of the plant and the user must insert this description for further use, *object-variables* are used to automatically abstract and store information from process variables. They are dynamically actualised.

Since the embedded information in an *object-variable* is obtained directly from the numerical signals after performing certain simple operations (with use of *abstraction tools*), sometimes external parameters are necessary. These

parameters must be supplied to each *object-variable* in order to provide the process knowledge concerning this variable. This kind of knowledge is necessary to distinguish certain variables of processes. For instance, a control variable of a level regulation process is qualitatively different from a DC motor speed control variable, but both could be acquired as a voltage in the range of 0 to 10 volts.

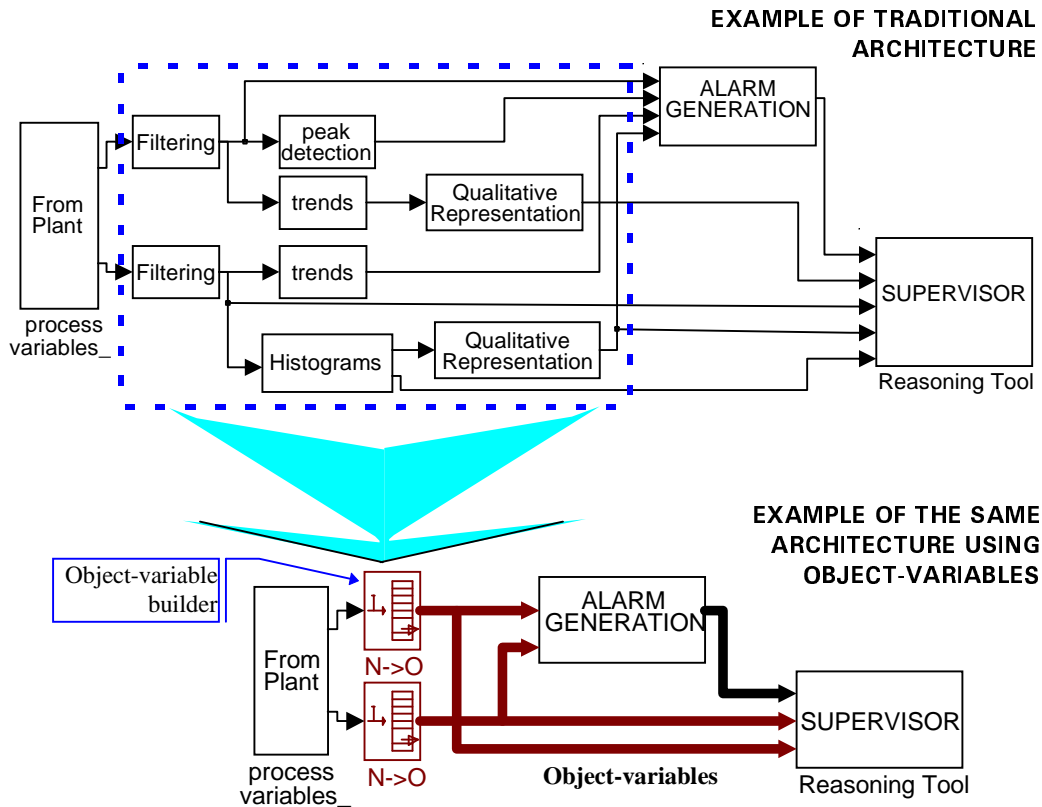


Fig. 5.2 Comparison of traditional implementation of a supervisory architecture and implementation using *Object-Variables*.

Implementation of *object-variables* in Simulink has been carried out within the framework of S-function blocks. Simulink blocks have been used to graphically represent the conversion of simple variables into objects (*object-variables*). Optional parameters can be supplied externally, using dialogue windows associated to each *object-variable*, as shown in “Fig. 5.3”.

The blocks, labelled as "N->O" in the figure, perform this conversion. Objects are built from numerical variable values according to the *object-variable* definition and supplied parameters. Therefore, the output of these blocks is not numerical data but it constitutes *object-variables*, and all information embedded in them is accessible for the connected blocks. The figure, represents how two *object-variables* are supplied to a diagnostic block.

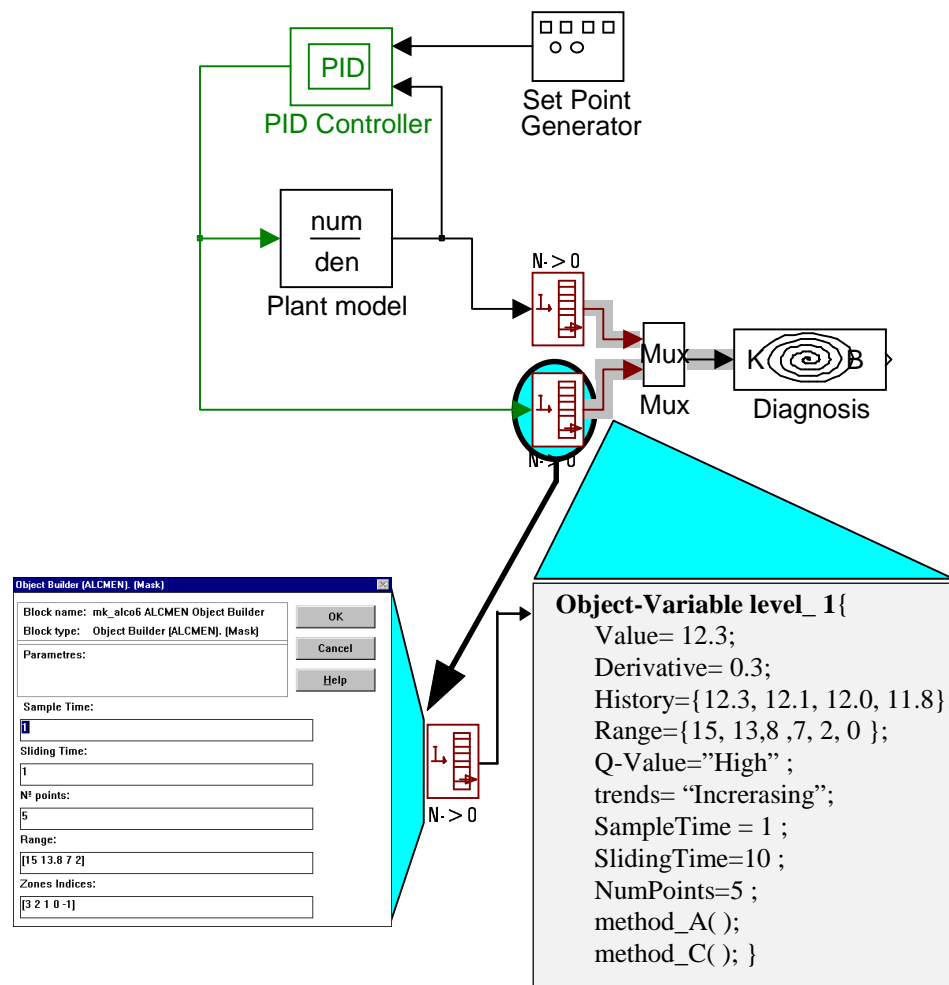


Fig. 5.3 *Object-variables* in Simulink allow to encapsulate information to be shared with other blocks.

The same encapsulation can be applied not only to process variables, but also to indices obtained from them or the combination of several variables. The aim is to supply a multiple representation of the same variable (process variable, index, ratio, and so on) with an easy access to facilitate

5.2.3. About objects in MATLAB-Simulink. Containers.

The idea of using objects in MATLAB was firstly put forward in [Maciejowski and Szymkat, 1994] introducing the use of ‘containers’. This work presents ‘containers’, which have some of the commonly accepted properties (but not all) of the ‘objects’, to represent object-oriented models. Those ‘containers’ are used to solve data management problems in a programming environment for control systems analysis and design, selecting MATLAB as a test-bed. The benefits of using ‘containers’ in CACE frameworks are also discussed in [Saifuddin, 1996].

The features of ‘containers’ common to objects are :

- Classes of containers can be defined,
- Properties of containers can be updated and queried,
- Containers can be interrogated about themselves,
- Data can be associated with methods .

After the work [Maciejowski and Szymkat, 1994], the evolution of MATLAB has always pointed to the idea of incorporating object structures. In fact, the new version (MATLAB 5.0) is supplied with five different classes of data and offers the possibility of creating new classes defining a MATLAB structure that provides data storage for the object [Matlab 1996]. These objects have some of the basic properties assumed for objects such as inheritance and overloading of the operators.

Another important characteristic of objects, is the encapsulation of methods and data in the same structure (maybe the most important feature of objects). The benefits of this property in a CACE framework are also discussed in [Saifuddin, 1996]. Despite of the importance of this feature, objects incorporated in MATLAB 5.0 do not provide this possibility. In fact, methods to deal with the desired objects must be defined apart from MATLAB objects, in separated files (*.M, MATLAB files) and stored in a specific directory. Due to this drawback Simulink 2.0 [Simulink 1996] deals only with numerical data at its input and output, reducing the applicability of this framework to numerical simulation and analysis. In case of applications to complex information processing, including elements of symbolic reasoning, the approach provided in the recent version of MATLAB is still far from satisfactory. Despite of improved graphical capabilities and the possibility of inheritance in some parameters (sample time, number of input and output), the use of Simulink as an Object oriented Computer aided control system design framework (OOCACSD) is still far, because of its poor capability of managing objects as variables.

The idea presented in this work is slightly different from the one of ‘container’ since real ‘objects’ are defined, using an external language, in MATLAB/Simulink. These objects are applied mostly to the description of variables supported by a Simulink block representation (graphical representation). The solution presented in this work is necessary for several tasks, such as supervision or diagnostic applications where tools need to operate on different kinds of data (qualitative, symbolic, numerical, logical, textual), representing the same process variable.

5.2.4. Classes of *object-variables*.

The use of *object-variables* is defined to encapsulate information (data and methods) into a data structure to be shared among different tools. It is assumed that an open architecture is used and new tools can be added into the framework. Then, new features may be required to describe system variables and *object-variables* must be endowed with new methods to supply them. Taking this into account, two possible solutions to modify the structure of *object-variables* are proposed below.

The first solution is a straightforward one. It consists in adding new features to the original *object-variable*, in order to obtain the capabilities required for any newly added tool. In this case, the data structure grows in size and some memory is likely to be wasted, even though not all features will be used but only some of them. This solution, although applicable directly, is not recommended in case of complex systems because of the great number of variables involved.

The second possibility is to define several *object-variables* with a simple common structure and particular methods to serve sets of tools with some similarities. This option allows to define more accurate *object-variables*, fitting the tool requirements in a more exact way; unfortunately in some cases a certain degree of incompatibility with certain tools may occur and more than one *object-variable* could be necessary to be used for representing several features of a process variable. Both possibilities are represented “Fig. 5.4”.

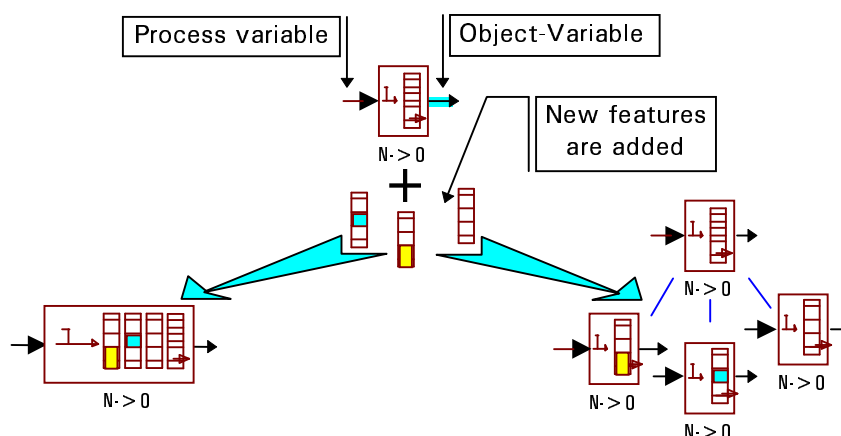


Fig. 5.4 Obtaining New *object-variables* from a basic structure.

The necessity of having several classes of *object-variables* is basically due to the necessity of integrating several tools for supporting the design of supervisory structures. The classes of the variables will be defined according to the specific requirements of the applied tools. “Table 5-1” shows the relationship among the

tools to be used to deal with abstracted information at several levels of abstraction and the kind of methods and data to be stored into the *object-variables*.

Object-Variables	low	← Abstraction degree →	High
Information	Measures	Numerical	Qualitative Knowledge
Data	Double, int, ...	Numerical types	Labels Facts
Methods	Signal processing	Numerical and abstraction algorithms	Inference engine
Tools	Instruments Filters	Fuzzy reasoning Simulation	Qualitative reasoning Rule-based systems

Table 5-1 *Object-variables* definitions according to the tools using such *object-variables*.

Several kinds of *object-variables* can be defined for working at multiples abstraction degrees. The proposal consists in declaring a hierarchy of *classes* capable of supplying adequate information to any tool. At this moment only two levels have been defined (See Fig. 5.5). The first is assigned to *object-variables* and to interface numerical process variables with qualitative reasoning tools. For this purpose several methods can be used (*abstraction tools*, presented in the next subsection). *Object-variables* can also be used as only qualitative variables without numerical attributes. The second level consist in a higher level structure, grouping *object-variables* and additional parameters and methods, called *facts*. These are used to interface a rule-based system avoiding different kinds of data as input. See subsection 5.5. *Facts* can be build directly without containing a complete *object-variable* inside. This means that they can be used as user input at run time to supply data to an ES.

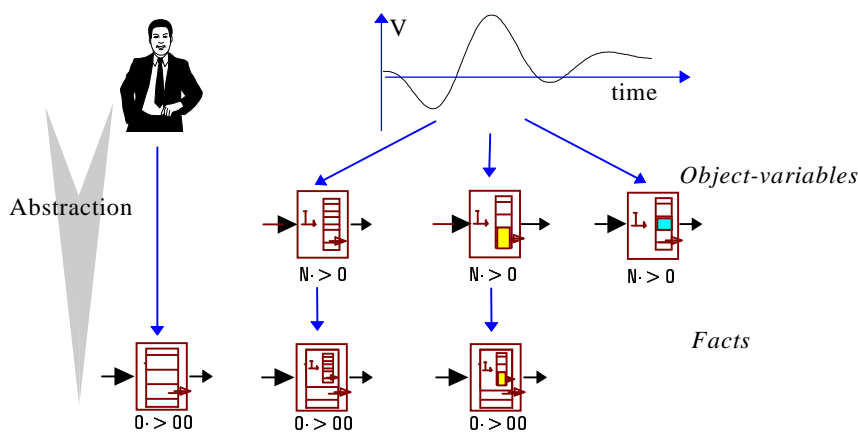


Fig. 5.5 Two hierarchical levels of *object-variables* have been defined.

The internal structure of these classes is defined according to OOP because of the possibility of embedding data and methods in the same structure, as is exemplified in Fig. 5.3.

5.2.5. Embedding objects into MATLAB/Simulink. Technical viability.

Due to the relevance of embedding objects into Simulink in respect to this work a brief explanation is added as technical note. Through the mechanism of S-functions defined in Matlab/Simulink, the user can add new general purpose blocks or incorporate an existing 'C' or 'FORTRAN' code into the simulation. The only constraint which must be taken into account is that the new code must be written with respect to the predefined structures or routines. These routines will be called by Simulink at each simulation step.

'SimStruct' [Simulink 1993][Simulink 1994] is the data structure used by Simulink designers to encapsulate the block's information. Each block in a Simulink representation has an associated 'SimStruct', accessed by Simulink to perform simulations correctly. Then, the information embedded in this data structure allows Simulink to know the parameters associated to this S-Function, as well as to access user defined routines.

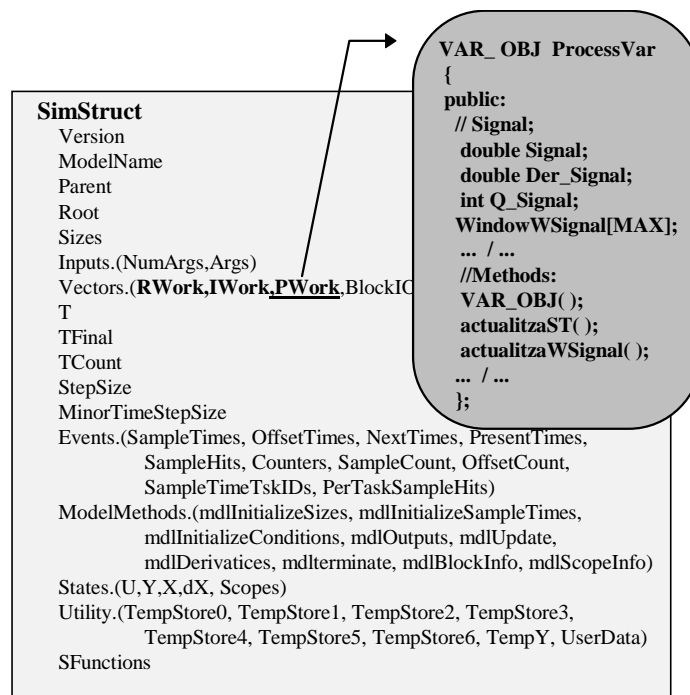


Fig. 5.6 Internal structure in Simulink.
Solution to embedded objects.

There are three fields where the user can save restricted information in the 'SimStruct' structure, only available for the object owner. These three fields are associated with work vectors that correspond to pointers to integers (int *), to doubles (double *), and pointers to void pointers (void **). In the 'SimStruct', see "Fig. 5.6", these fields are labelled as '*vectors.RWork*', '*vectors.IWork*' and '*vectors.PWork*'. Therefore, the only method to embed external information into Simulink 'SimStruct' consists in using these working vectors.

By taking advantage of this capability, it is possible to associate an object to Simulink blocks by saving a pointer (to this object) in the '*vector.Pwork*' field in its 'SimStruct'. Moreover, to each Simulink block there can be an object associated [Melendez et al., 1996c][Melendez et al., 1996b].

In order to assure correct work of the components together, some further constraints must be taken into account. The source file for getting a .MEX file (as a result of a compilation of an S-function), is structured according to a template structure defined by Simulink developers. This structure is composed of routines where the user can insert his own code. In these routines of the .MEX file under design the following advice must be taken into account:

#include "obj_def. h"	Class definitions and object access methods can be defined in an include file.
mdlInitializeSizes()	A pointer vector will be used and specified using the command 'ssSetNumPWork(S,1)'.
MdlInitializeConditions()	Memory space must be allocated, to be used by the object and its pointer.
mdlOutputs()	Methods could be activated to access the desired fields into this routine.
mdlUpdate()	To update and fill the predefined fields of the object.
mdlTerminate()	Remember to free allocated memory.

Table 5-2 Structure of a C written S-function.

The fact of working with objects implies using an object oriented compiler, and at the same time, access to data pointers (C++ compiler) must be allowed. In this case, Watcom 10.5 C++ compiler was chosen.

5.3. Abstraction tools.

In the previous subsection, the use of OOP in the definition of variables has been introduced in order to give a *multi-aspect* representation to the process variables. *Object-variables* have been presented for encapsulating data and methods used to obtain these multiple representations. But, what kind of

information is needed from process variables and which methods can be used with this purpose?. These topics are discussed below and several algorithms are proposed. Thus, *abstraction tools*, also called *abstractors*, refer to such algorithms that can be encapsulated into *object-variables* to supply qualitative representation of process variables.

5.3.1. Significant information from process variables.

Process variables, coming from real data (sensors and controllers) or simulation (analytical models) are mainly thought to be numerical data. This is the kind of information used for control loops, monitoring, alarm generation and fault detection in model-based systems. On the other hand, knowledge-based systems (used in fault detection, diagnosis and supervision) use inference methods for reasoning about both numerical (fuzzy reasoning) and qualitative information (qualitative reasoning). This qualitative data can be abstracted from the process, provided by engineers or supplied from a qualitative simulator or knowledge-based system.

The particular form of abstract data obtained from process variables depends on the process and on the application to be developed. Therefore, different techniques for obtaining significant information (numerical, qualitative, symbolic and logical) from process variables could be used. Moreover, they could be used by qualitative reasoning tools as numerical to qualitative interfaces. See for example Fig. 5.8. The significant information which can be obtained from signals analysis, is :

- Numerical information (additional) :
 - + Derivatives.
 - + Trends.
 - + Deviations.
 - + Distances with respect to some predefined shapes.
 - + Relative extreme.
 - + Landmarks (significant points and alarms).
 - + Indices and associated parameters (damping factor, areas, overshoot, ...)
 - + Frequency contents.
 - + Integrals.
 - + Certain mean values.
- Qualitative or symbolic description of signals (numerical to qualitative interface) :
 - + Labels.
 - + Episodes.
 - + Qualitative level of signal.

- + Qualitative Trends.
- + Qualitative deviations, and so on.

These descriptions and additional information, such performance indices or ratios, may be obtained on line with data acquisition. Moreover, their combination could be applied in event generation or for obtaining trends of certain parameters, for example. A wider enumeration and description can be found in [Rakoto-Ravalontsalama N., 1993]. Several qualitative representations of signals can be obtained with different methods as is depicted in Fig. 5.7 and Fig. 5.8. The choice of one or the other depends on the tool that must use this information.

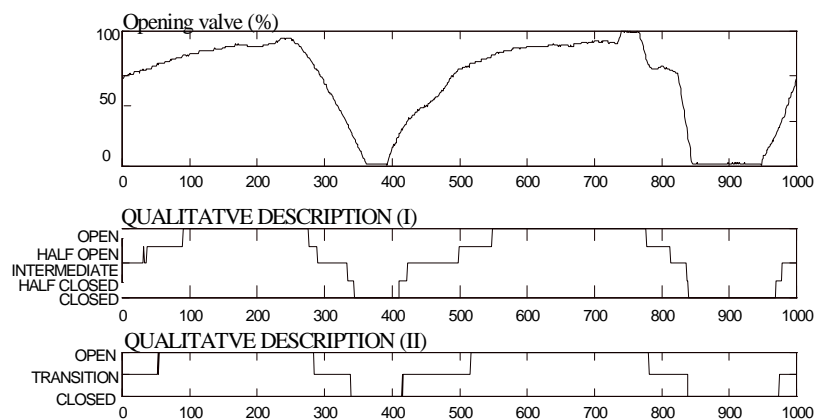


Fig. 5.7 A simple division in zones of the amplitude space can give several qualitative representations of a signal.

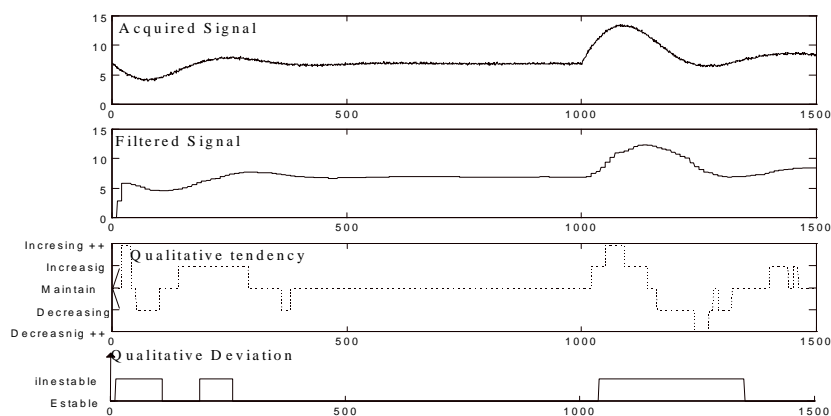


Fig. 5.8 Qualification of filtered signals in crisp zones could be used to provide with qualitative tendency and qualitative deviation.

Further, elaborated numerical information is obtained using some classical signal processing methods:

- *Derivatives* permit to know instantaneous changes in dynamics of variables.
- *Polynomial regression* could be useful for obtaining an analytical description of variables before performing some additional calculation.
- *Filtering* is applied not only for smoothing signals, but also for dynamics isolation.
- *Fourier transform* for frequency analysis of dynamics of variables can help in tuning and selecting parameters.
- *Wavelet transform* gives a representation of signals at different scales, in both frequency and time.
- *Statistical signal processing* methods are used to capture statistical features, such as mean value, variance and standard deviation, correlation analysis, and so on.

Other, non-classical methods can be used to obtain a qualitative description of signals. As a result of such application, a label (linguistic description) or a set of labels of signals is obtained. Thus, a label is used to describe in a symbolic way signal behaviour during a certain time period. Some characteristic labels cover particular general changes of the signal level, which are distinguished by the description of selected *episodes*. A more complex signal characterisation can also be formed in this way.

- *Triangular episodes* is another representation of signals that allows time representation at several degrees of resolution.
- *Amplitude splitting*, is used to associate qualitative labels to representative zones of the operative range of a variable.
- *histogrames* are used with the same purpose, but with low pass filtering capabilities.
- *Pattern matching* gives a distance related to expected shapes of trajectory behaviour.

These methods could be combined to obtain a more elaborated description of the process behaviour according to the process variables. Fig. 5.9 shows a more extended classification of numerical methods to be used for obtaining qualitative representation from numerical data. The main division is given between frequency based and temporal methods according to the kind of features to extract. Note that the majority of methods taken into account are based on not only in a single sample but the history of signal. For example, frequency based methods, such as FFT or Wavelet based need a representative number of samples, while filtering methods could be implemented with less samples. Signal history is also needed in window-based (histogrames, regression, pattern matching, ...) temporal methods or triangular representations.

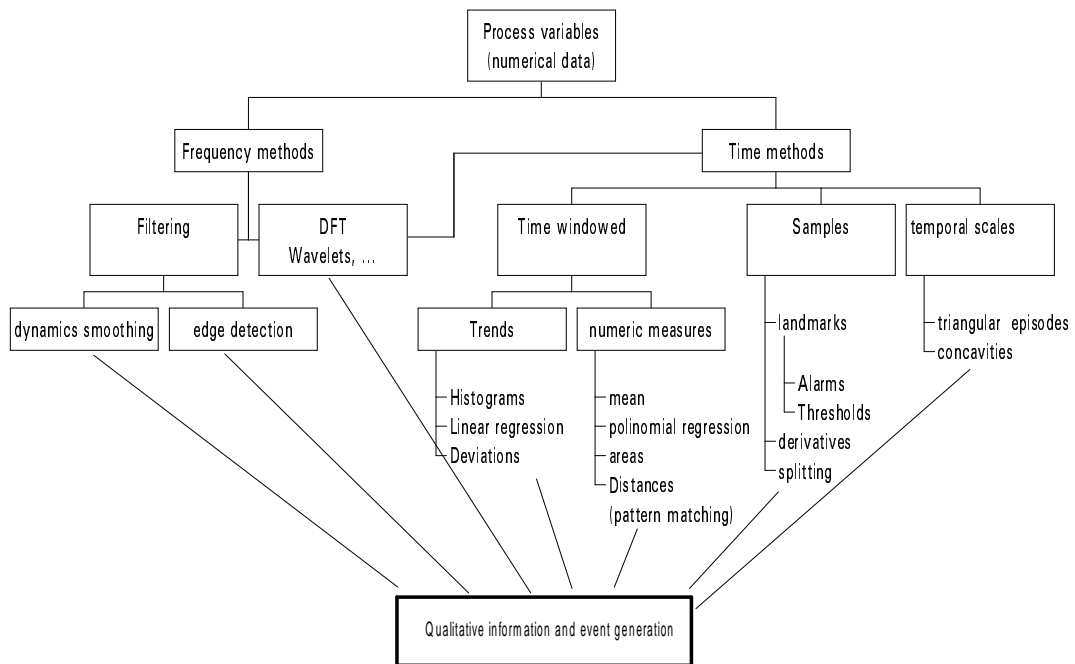


Fig. 5.9 Classification of abstraction methods.

5.3.2. A taxonomy/selection of abstraction tools.

Since the currently applied sensors provide only numerical data, some tools must be supplied to infer qualitative appreciation of the process behaviour from these measures. Then, the necessity of tools designed to provide external components with significant information is clear. These tools are called *abstractors (abstraction tools)*.

Abstractors are useful in process supervision in three different ways: The first one is to perform the numeric to qualitative conversion in order to provide expert systems or reasoning tools with handily significant information and to reduce the information overload. The second, from the analysis point of view, to avoid meaningless information from measured signals and other variables, giving visual representation of the process dynamics through *abstractors* like trends, deviations, tendencies, and so on. Third, they are used to form the symbolic information, constituting the input for knowledge-based inference tools, such as ES and qualitative reasoning tools as is explained in following subsection.

Moreover, it is clear that sensors provide only dated samples of a process variable. Thus, interpretation of acquired values according to process behaviour must be done by taking into account all additional information that can be supplied to obtain the necessary knowledge of interest about process behaviour.

5.3.3. Kinds of abstractors.

As pointed in [Aguilar-Martin, 1993] supervision and diagnosis tasks must include expert knowledge and reasoning about qualitative information. Therefore, several methods have been proposed to obtain qualitative representation of situations using process variables to generate qualitative information for these tools for supervision, detection and diagnostic tasks [Dorf R.C., 1993] and [Ganz, Kolb and Rickli, 1993].

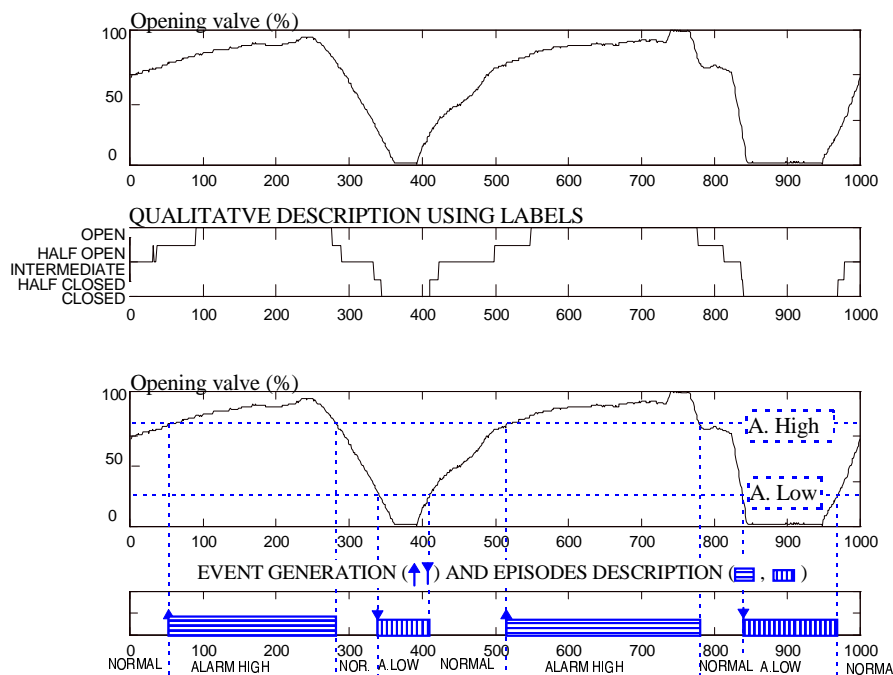


Fig. 5.10 Different qualitative representations for the same process variable using qualitative labels, event generation and episodes.

Qualitative description of signals is inherent to the method applied to signals. Thus, different methods applied to the same signal can supply distinct qualitative information. Moreover, these qualitative representations of signals can be supplied using several temporal references (synchronous or asynchronous references). According to the time sequence, this text considers qualitative information obtained from signals is divided into three categories (Fig. 5.10):

- *Qualitative labels* are labels associated to the level of signals and actualised at every sampling time or every selected time period.
- *Event generation*. It is the description of specific situations. These are asynchronous results related to process variable evolution used to identify that a *landmark* is reached or to fire simple *alarms* when applied directly to that task.

- *Episodes*, or historic description of signals as sets of labels, used to define a specific behaviour during an interval or period of time. An *episode* is defined by a starting and an ending time point and a *event*. Thus, the same label can be applied to *episodes* of different duration.

Different techniques for obtaining qualitative information from process variables are described in the following paragraphs. Some of the most representative include the following ones :

- Qualification of filtered signals: Signal filtering is a straightforward way of isolating process dynamics from variables. Filtering, and its later qualification, allows to obtain global trends as for example, the description based on qualitative representation of *tendency* and a *deviation* from this *tendency*. This was employed in [Melendez et al. 1995] to provide an expert system with qualitative information. In this case *qualitative labels have been used*. This representation is reproduced in Fig. 5.11.

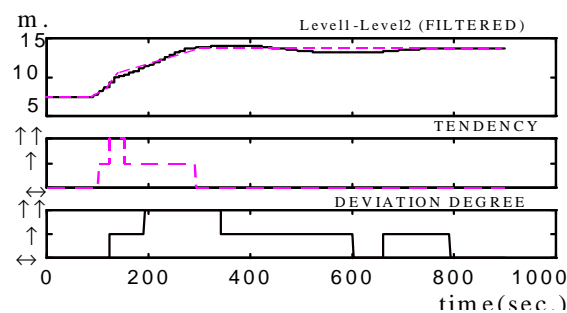


Fig. 5.11 Qualification of filtered signals in crisp zones could be used to obtain a representation of signals in terms of qualitative tendency and qualitative deviation degree.

- Histogrames. Histogrames [Rakoto-Ravalontsalama N., 1993] perform a segmentation of the amplitude spaces of measured signals during a period of time (temporary window). [Sarrate R. et al., 1995] uses histograms to evaluate the signals in their evolution in a sliding time window. Several histogram indices could be used in supervisory applications. For instance, *dominant mode* is a simple way for obtaining a *qualitative label* corresponding to the principal zone used in the time interval corresponding to the chosen time window. Other indices, such as *dominance degree* or *entropy*, offer numerical information to be added to *qualitative labels*. Histogrammes have some characteristics of low pass filter, which frequency cut corresponds to the length (or memory) of the time window, reducing noise effects. In consequence, a time delay is expected in the response. This technique can be directly applied to signals before or after filtering. Fig. 5.12 shows the use of a dominant mode to represent qualitative states of a noisy signal.

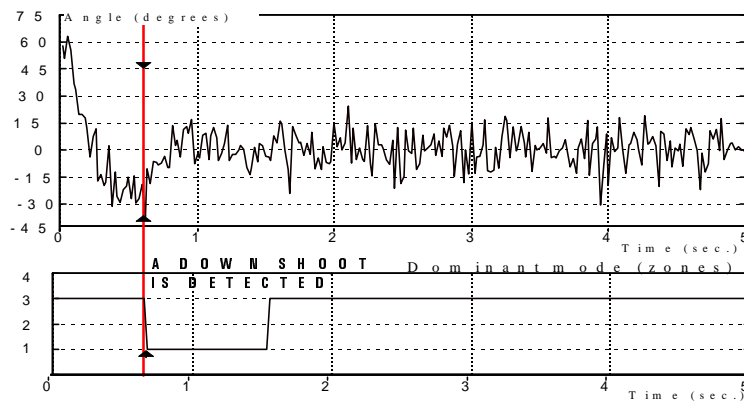


Fig. 5.12 Use of histograms (dominant mode) for peak detection in a noisy signal.

- Triangular representation. [Bakshi B.R. et al., 1994], [Ayrolles, 1996] and [Cheung and G. Stephanopoulos, 1990]: Triangular representation is a qualitative description of measured signals behaviour. Using singular points of the signals (maxima, minima and inflexion points) and time intervals between these points, triangular episodes can be represented. These episodes give information about the tendency and curvature of signals. If triangles are grouped in trapezes, then the representation at different temporal scales is obtained (See “Fig. 5.13”). This kind of representation is used in the multi-scale qualitative description of trends. [Colomer et al. 1997] add numerical information to this episodes for describing areas, lines, slopes and so on.

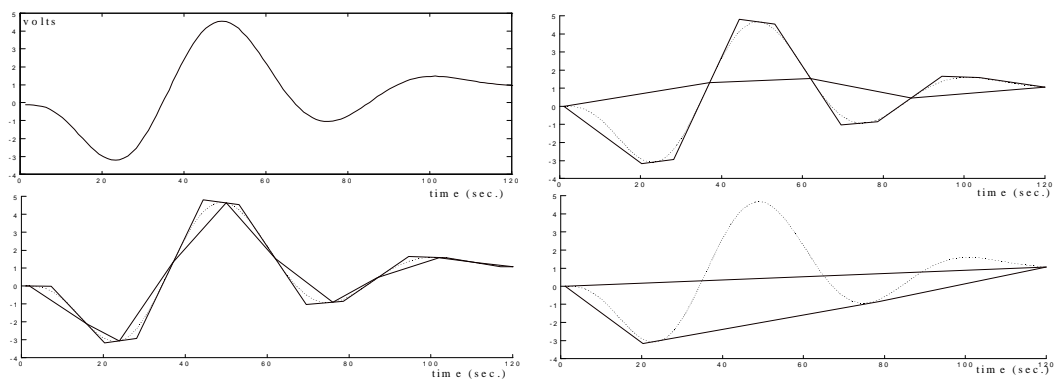


Fig. 5.13 Triangular episodes offer a temporal multi-scale representation.

- Wavelet transform [Bakshi B.R. et al., 1994]: Wavelet transform is a signal decomposition technique in temporal and frequency domains with various resolutions. In this way, several temporary and frequency scales of representation can be achieved. This allows a description of measured signals to contain quite different dynamics. These signals decomposition could be very useful for supervisory processes to study

the most distinguished signal features and dispose others without interest, as suggested in [Bakshi and Stephanopoulos, 1994].

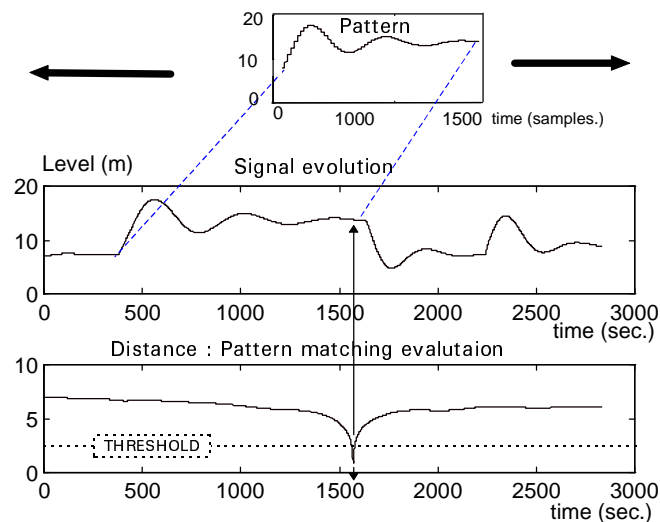


Fig. 5.14 Pattern matching

- Pattern matching, could be used to detect special situations (for instance in fault detection) comparing signal evolution with patterns extracted from the previous faulty situations. A distance is used to define the degree of coincidence (“Fig. 5.14”). Also correlation methods can be applied with this goal. As a result, *events* are fired when a threshold in the distance or correlation is surpassed.
- Linear (or polynomial) regression, of signals in a period of time allows to represent tendencies and classify signals behaviour according to polynomial parameters and properties.

Despite of the utility of such tools for obtaining qualitative representations of process variables, the main drawback resides in the election of the adequate method. Moreover, in the majority of such algorithms, it is necessary to tune some parameters as crisp limits for qualitative zones, sampling time, orders, number of samples (history length) and so on. For an adequate election of these *abstraction tools*, sometimes it is necessary to know how the algorithm works or the adequacy for obtaining specific information. Next table treats to resume some dependencies among algorithm characteristics, process and abstracted information.

For example, when using pattern matching techniques it is necessary to have a register of signals stored in a previous failure. Or the use of filtering techniques is associated to the presence of variations in the evolution of signals. The richer the signal, in terms of frequency, the better results are reached. On the other hand, all the algorithms that operate with signals history are submitted to a delay in the response time, but their fiability increases.

	Filtering + qualification	Histogrames	Triangular representation	Wavelet transform	Pattern matching	Polynomial regression
Knowledge about method	High	Medium	Low	High	Distance to be used	Degree
Knowledge about process	High	Medium	Low	Very low	Special situations	Process independent
Configuration	Difficult	Easy	Not needed	Easy	Not needed	Not needed
Robustness	Low	High	Relative	High	Low	
Process limitations	Oscillations	Slow	Oscillations	No limitations	Noise	Number of points
Abstracted information	Tendency, deviation degree....	Dominance, dominant mode and entropy	Tendency and convexity at different scales	Behaviour at different scales	Coincidence degree.	Relative to polynomials
Main performances	Discrimination between tendency and deviation	Reliability	Qualitative representation at different time scales	Numerical representation of behaviour at different scales	In fault situations with known dynamics.	Analytical description

Table 5-3 Comparison of some abstraction tools.

“Table 5-3”, extracted from [Melendez et al. 1996a] , resumes the main features of these *abstractors* according to the facility in configuration, the dependency with process dynamics and the information provided from numerical signals. These features are obtained from the experience and the use of these algorithms with different types of signals with the purpose of obtaining qualitative representations of them.

5.3.4. Abstraction tools and *object-variables*.

Abstractors are the proposed tools or algorithms to interface purely numerical process variables and components based on more abstract expert knowledge, providing reasoning tools for the generation of qualitative representation of variables. This interface is different depending on the supervisory strategy to be developed and the expert knowledge base. Taking benefit of the object-oriented approach presented in the previous subsection for describing process variables as *object-variables*, abstraction methods are proposed to be encapsulated in the object structure as internal methods to obtain significant information for the qualitative representation of process variables. Graphical representation of *object-variables* as individual blocks are used to identify each process variable and the abstracted information supplied by the embedded algorithms. Then, a set of blocks representing *object-variables* with the same internal structure, can be associated to different abstraction methods. The only difference resides in the method or methods implemented and the information obtained and stored in them.

5.4. ALCMEN, A representation language.

ALCMEN (Automaticians Language for Causal Modelisation for Expert kNnowledge) was conceived by [Aguilar-Martin, 1991a] as a language for representing and dealing with imprecision and uncertainty in algebraic and differential equations. The ultimate aim of the language is to facilitate communication between process engineers (domain experts) and control engineers. In ALCMEN, algebraic and differential equations have a representation for qualitative values. Because of the basic features in knowledge representation and capabilities for dealing with simple qualitative relations, ALCMEN has been chosen to be implemented in MATLAB/Simulink as a block based ToolBox for dealing with qualitative data. The objective is to implement this ToolBox with capabilities of dealing with qualitative data, embedded into *object-variables* by means of simple connections between blocks. The goal is to use *abstraction tools* as interface numeric to qualitative and ALCMEN to represent simple qualitative dependencies between variables when analytical relations are not known at all. ALCMEN has been tested as a tool for developing qualitative observers as is explained in [Melendez et al. 1996a], and reproduced in chapter 6.

5.4.1. Fundamentals of ALCMEN.

The principal idea of ALCMEN consists in graphical knowledge representation with blocks having the capability of representing a relationship among variables with imprecision. Basic operations in ALCMEN are thought to provide a simple qualitative operator between variables. Therefore, numeric to qualitative interfaces are needed. For this purpose *abstraction tools* can be useful to give the adequate representation for each design. Despite some simple algebraic operations available in ALCMEN, it is not defined as a qualitative algebra but only as a mechanism for knowledge representation and intuitive relation of qualitative variables.

In ALCMEN *variables* are thought to be structured as objects, encapsulating attributes and methods. These attributes are *type* (numerical, qualitative or mixed), *range* (set of possible values), *subsets of values* (desired or usual values, *landmarks*, and other useful information). The methods are basically conceived to perform operations or relationships between these variables and to obtain qualitative representations from numeric values. Blocks are used to graphically represent causal graphs where the input of blocks are *variables*, representing causes and the output are the effect *variables*.

ALCMEN variables allow both numerical and qualitative values in the same representation according to the following descriptions :

- *Numerical variable*: Values are given by a real number (Notation: x).
- *Lexical domain, S_{is}* : A set of ordered labels, describing symbolic (or qualitative) values, indexed by correlative integer indices. Each label is associated to one of these indices from the lowest, i , to the highest, s :

$$S_{is} = \{(label_{[0]}, i), (label_{[1]}, i+1), \dots, (label_{[k-i]}, k), \dots, (label_{[s-1-i]}, s-1), (label_{[s-i]}, s)\}$$

- *Lexical variable, X* : It is a qualitative variable. The set of possible values is defined in the *lexical domain, S_{is}* . Thus, a *lexical variable* in ALCMEN is represented by a pair formed by a *label*, $\langle x \rangle$, and the corresponding integer index, n_x : (Notation : $X = (\langle x \rangle, n_x)$).

Qualitative representation of magnitudes in ALCMEN is given by indexed *labels* or *lexical variables* (in the ALCMEN nomenclature). *Labels* are ordered using correlated (or corresponding) integer indices. The mechanism used as numerical to qualitative interface is called *filtering*, according to ALCMEN nomenclature. It basically consists in splitting the amplitude space of a *numerical variable* into crisp zones, see Fig. 5.15. The indexed set of *labels* associated to these zones is called the *lexical domain*. Then, the *filtering* operation associates a *numerical variable*, with a *lexical variable* from its *lexical domain* at any time. Thus, given a *lexical domain, S_{is}* , formed by a set of qualitative *labels* and its associated indices, k , ordered from the lowest, i , to the highest, s , the *filtering operator* of numerical variable, x , is described by the operator $F(x/i, s)$, (indices i and s , separated by a slash from the numerical variable, x , are the inferior, i , and superior, s , indices in the lexical domain used).

Given the *lexical domain S_{is}* :

$$S_{is} = \{(label_{[0]}, i), (label_{[1]}, i+1), \dots, (label_{[k-i]}, k), \dots, (label_{[s-1-i]}, s-1), (label_{[s-i]}, s)\} \quad \text{Eq. 5-1}$$

The *filtering* operator is applied to a numerical variable, x , to obtain the corresponding *lexical variable, X* :

$$X = (\langle x \rangle, n_x) = F(x / i, s) \quad \text{Eq. 5-2}$$

So, *ind()* and *lab()* operators are defined to obtain n_x and $\langle x \rangle$ from the *lexical variable, X* :

$$n_x = ind(X) \quad \text{Eq. 5-3}$$

$$\langle x \rangle = \text{lab}(X) \tag{Eq. 5-4}$$

Filtering is defined to be applied to numerical values. Then, the index, n_x , of the lexical variable, X can also be *filtered*. When an index of a *lexical variable*, X , defined in a *lexical domain*, S_{is} , is filtered in the same *lexical domain*, the same *lexical variable*, X , is obtained. Thus, if a *numerical variable*, x , is filtered to obtain the *lexical variable*, X , with index n_x :

$$X = (\langle x \rangle, n_x) = F(x / i, s) \tag{Eq. 5-5}$$

As a result of filtering this index n_x , the same *lexical variable*, X , is obtained:

$$X = (\langle x \rangle, n_x) = F(n_x / i, s) \tag{Eq. 5-6}$$

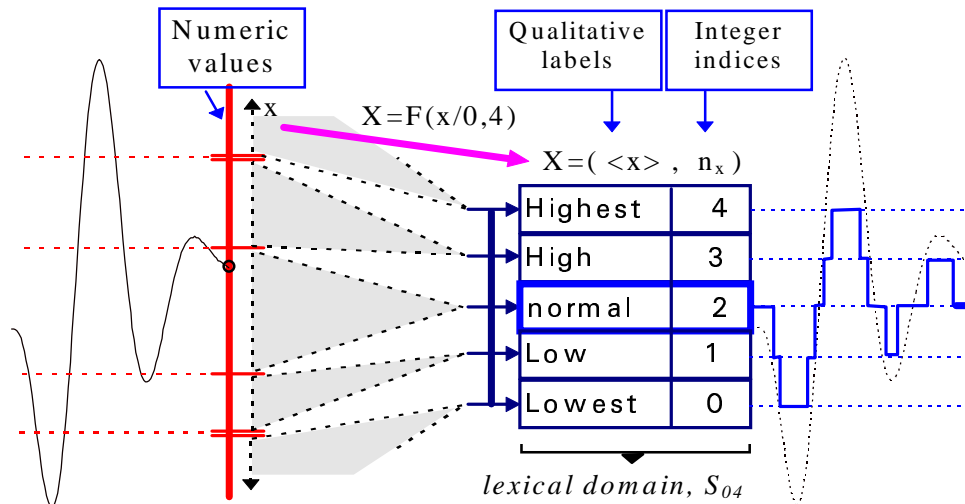


Fig. 5.15 Numeric to qualitative interface

Filtering is the simplest way of obtaining qualitative representation of process variables, although other mechanisms can also be applied to associate qualitative labels to numerical variables. In fact, the previously presented *abstraction tools* are thought to be used as numeric to qualitative interface. The only restriction, is to use indexed labels in the resulting qualitative representation. Using this kind of representation, ALCMEN language permits to perform a vast set of operations and to represent different types of relationships. It is not defined as a complete algebra, but it is intended to deal with simple and robust relationships between qualitative variables. The aim of ALCMEN is to provide a mechanism of comparison, representation of relationship, and simple operations.

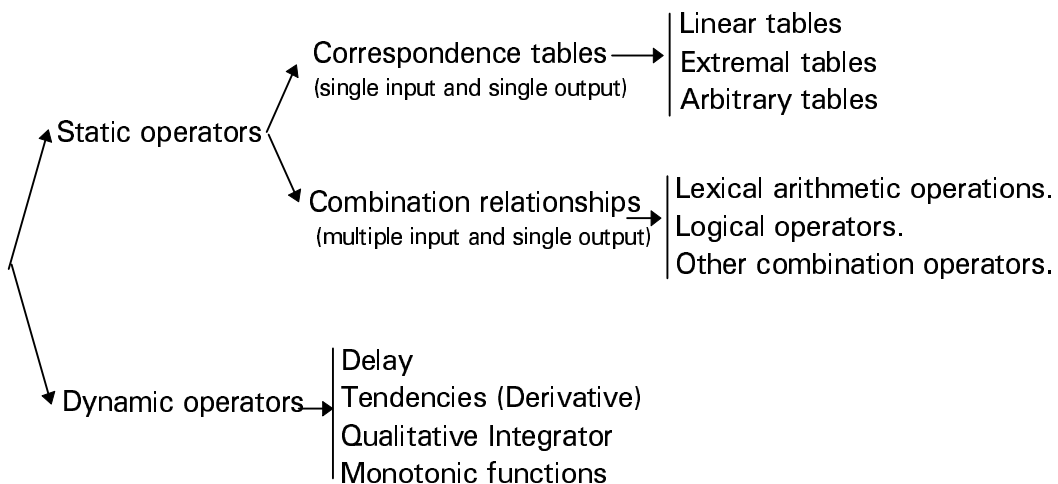


Fig. 5.16 Classification of ALCMEN operators.

The kind of operators admitted are grouped, as depicted in Fig. 5.16, in two major sets : dynamic and static operators depending on the time dependence. Dynamic operators involve not only the current value of the variable, but also the previous values. This forces ALCMEN to be defined in the discrete domain. On the other hand, static operators are evaluated at any sample time using the current input to deduce the output. Implementation of static operators is performed by the use of tables. In the following subsections these operators are described.

5.4.2. Static operators

Lexical Arithmetic operations

- Lexical difference, Θ , between two lexical variables, $X=(\langle x \rangle, n_x)$ and $Y=(\langle y \rangle, n_y)$, in the same *lexical domain* S , is defined as :

$$X\Theta Y = F(n_x - n_y / i, s) \tag{Eq. 5-7}$$

The result of *filtering* the difference of indices in a *lexical domain*, S_{is} , is always an index in S_{is} . The *lexical difference*, is conceived for establishing comparisons between qualitative variables. Then, the goal is not to calculate a difference from the arithmetical point of view, but to see it as a measure of the error or comparison between two variables. Fig. 5.17 shows the use of this operator in feedback systems, as it is done in numerical systems.

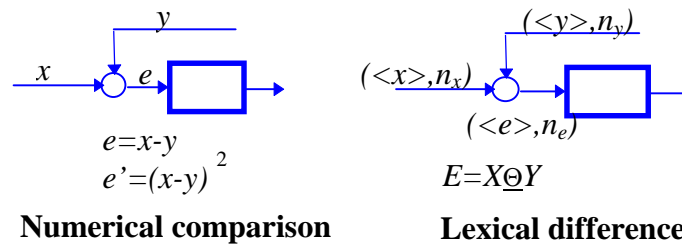


Fig. 5.17 Lexical difference is a way for comparing two variables.

- Lexical addition, \oplus , is defined between a lexical variable, $X=(\langle x \rangle, n_x)$, and an integer number, m , by using the indices of the lexical variable. The purpose of a lexical addition is to increase or decrease the qualitative variable m indices in the lexical domain. Thus, the maximum and minimum value at the output of a lexical addition is limited to the inferior and superior limits of the lexical domain.

$$X \oplus m = F(n_x + m/i, s) \quad \text{Eq. 5-8}$$

The generalisation of the addition for a *lexical variable*, $x=(\langle x \rangle, n_x)$, and a real number, y , is obtained by filtering the corresponding index of the *lexical variable* and the real number:

$$X \oplus y = F(n_x + y/i, s) \quad \text{Eq. 5-9}$$

Linear and Extremal tables

A lexical function between X and Y is defined as a mapping between the domain of X and Y , so that any value of X has a correspondent value in Y . All these functions are simply described using correspondence tables.

- Linear function (gain), is defined as follows for a pair of arguments, a lexical variable $X=(\langle x \rangle, n_x)$ and a real number, K (gain),

$$Y = X \otimes K = F(K \cdot n_x / i, s) \quad \text{Eq. 5-10}$$

It must be observed that the input and output set of indices are the same because X and Y are defined with equal *lexical domain*.

Examples: The representation of the linear function gain is done by means of an input/output table in Fig. 5.18. In these tables, cells filled with the symbol \mathbf{x} represents the relation between input and output, while symbol $\mathbf{0}$, represent not permitted output for the correspondent input. It is immediately deduced that when $K=1$, then $X=Y$. Tables can also be used to represent gain given by real numbers. For example, gain $K=0.5$ is represented in the same figure.

Lexical domain					
label	absence	small	normal	great	excessive
indices	0	1	2	3	4

GAIN = 0.5		Input					
		0	1	2	3	4	
Output	0	x	x	0	0	0	
	1	0	0	x	x	0	
	2	0	0	0	0	x	
	3	0	0	0	0	0	
	4	0	0	0	0	0	

GAIN = 1		Input					
		0	1	2	3	4	
Output	0	x	0	0	0	0	
	1	0	x	0	0	0	
	2	0	0	x	0	0	
	3	0	0	0	x	0	
	4	0	0	0	0	x	

GAIN = 2		Input					
		0	1	2	3	4	
Output	0	x	0	0	0	0	
	1	0	0	0	0	0	
	2	0	x	0	0	0	
	3	0	0	0	0	0	
	4	0	0	x	x	x	

Fig. 5.18 Linear function. Gain is from left to right K=0.5, K=1, K=2.

- *Extremal functions.* Minima or maxima functions can be described as associative relationships between the input and output of a block.

Definition : A *normoide function*, $N(n)$ is a real function defined for the indices (n) of a *lexical domain*, S, in another *lexical domain* P, with the properties:

- $(\langle 0 \rangle, 0) \in P$ (the lexical domain P contains the label $\langle 0 \rangle$ and the index associated is the integer 0,
- $N(n)$ is semi-positive real,
- $N(n)=0$ for only one value of $n = n_o \in ind[S]$.

A *minimal lexical function* is one given by:

$$Y = E(X) = F(n_e + N(n_x) / i, s) \tag{Eq. 5-11}$$

$n_x \in ind[X]$

where the index n_e corresponds to the minimum of the E(X) function obtained when $n=n_o$ by lexical addition to $N(n_o)$. Analogously, a *maximal lexical function* is obtained with the equation :

$$Y = E(X) = F(n_e - N(n_x) / i, s) \tag{Eq. 5-12}$$

$n_x \in ind[X]$

Examples : The tables in Fig. 5.19 represent, these extremal functions for two different cases of maximal and minimal relations. Minimal and maximal index is given by the cell marked by n_o . The first table gives a minimum value for the input equal 2, and the output result is $n_o = 1$. The same dependency is given in the next minimal table where the input equal 1 gives the minimum output index $n_o = 0$. Similar explanation can be given by a maximal relation.

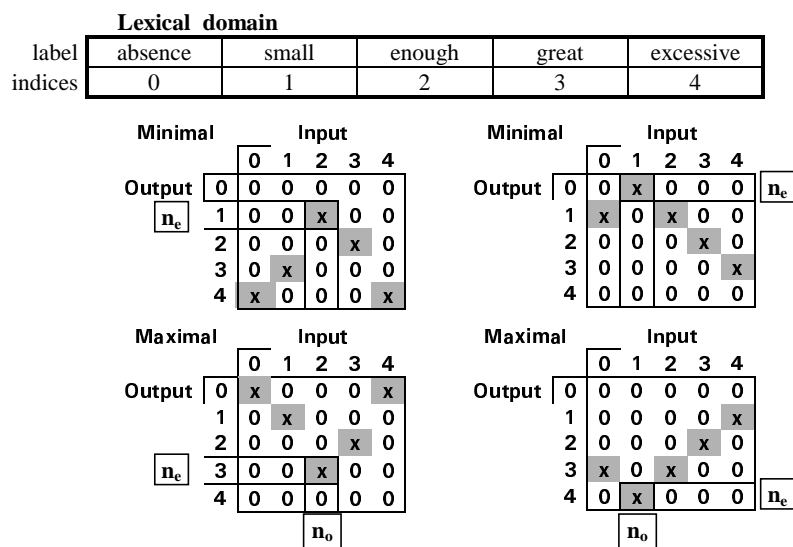


Fig. 5.19 Extremal functions.

Combination Operators:

Cause-effect qualitative relationship with more than one input has a representation in ALCMEN using tables (or cubs or hypercubs according to the number of inputs). The indices corresponding to inputs of tables are situated in the axes and the output is selected by the cross-point of the co-ordinates. The same tables used for logical operators are applied to define other qualitative operations between variables (norm, distance, ...) similarly to arithmetic operators.

Example : Conjunction operators (AND) for two inputs defined with a *lexical domain* of five labels. This AND is defined as the minimum of both inputs.

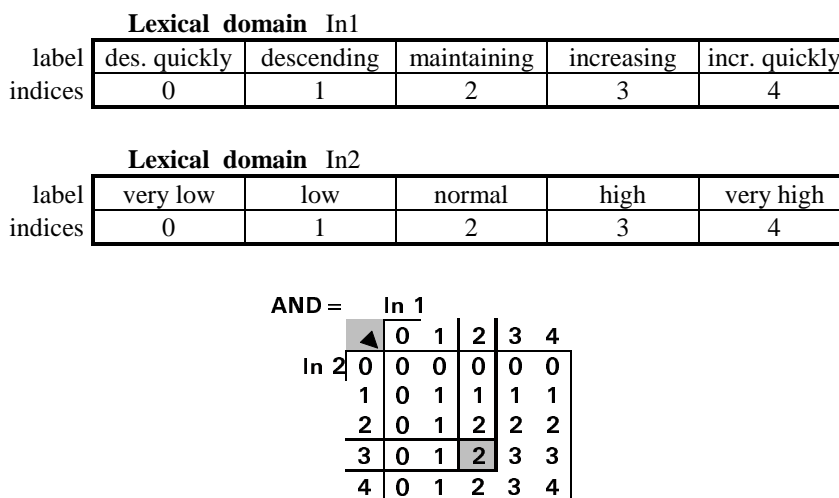


Fig. 5.20 Combination operator AND.

5.4.3. Dynamic operators (time depending relationships).

- Delay (∇ or elementary memory). Assume that a temporal sequence of indices corresponding to the evolution of a qualitative variable (X) is given. The operator ∇ gives as output the value of the input in the previous sample time (X_{k-1}). The following equation shows this property.

$$X_{k-1} = \nabla X_k \quad \text{Eq. 5-13}$$

This is the basic operator used to describe temporal relationships as tendencies and other monotonic causal relationships.

- Tendency (DX), of a qualitative variable is obtained applying the qualitative difference between one sample and the previous sample :

$$DX = X_k - X_{k-1} = X_k - \nabla X_k \quad \text{Eq. 5-14}$$

5.4.4. Qualitative representation of numerical signals.

This section discusses briefly the problem of splitting the amplitude space into crisp zones and assigning them qualitative labels (signal level abstraction). The selection of limits of these zones is clearly dependent on the particular application and on the origin of the signal. As an example, a first order response is studied and qualified in five zones ($Z1$ to $Z5$). Two different criteria have been used to decide about zone limits (Fig. 5.21):

1. Equi-spaced zones. Limits are selected dividing the range of signal into five zones of equal amplitude.
2. Limits of zones are selected at amplitudes corresponding to systems response at multiple of time constant.

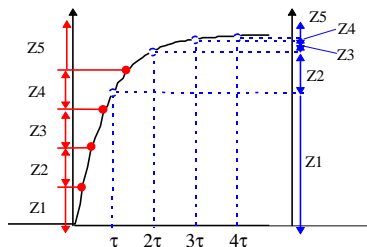


Fig. 5.21 Selection of limits for splitting the signal range in five zones.

The shape of the qualitative variable obtained in either case for a first order continuous system is different. The first case follows the first order system

dynamics while the second option produces a linear change of zones and more emphasis is put on how the permanent state is reached.

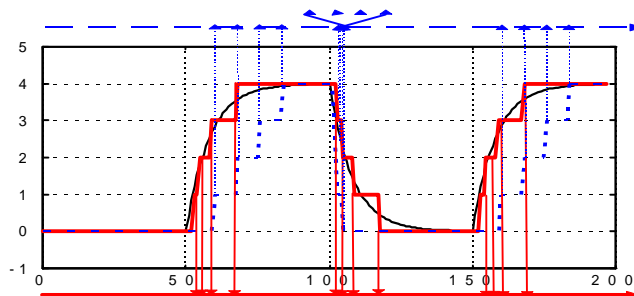


Fig. 5.22 Event generation for a first order system according to selected crisp zones in Fig. 5.21.

The importance of numerical to qualitative interfaces is clearly application dependent as is represented in Fig. 5.22. The same system is qualified using different a crisp due to the necessity of observing the amplitude evolution or time dependency of the response (Fig. 5.21).

5.4.5. ALCMEN and qualitative reasoning.

Nowadays, qualitative reasoning is one of the fields of increasing interest in AI and new paradigms are constantly added. The advantage of qualitative techniques is the way of managing rough and imprecise information, but as a result of using these techniques for reasoning rough and imprecise results are obtained. Therefore, qualitative techniques would be really useful when mixed with other techniques for reasoning, and not used as an isolated tool. Some suggestions have been pointed in [Koch, 1993]. In order to improve the results obtained by qualitative techniques in reasoning about system behaviour, one can also perform complementary operations, such as for example:

- Adding quantitative information.
- Exploiting advanced mathematical methods, like the qualitative methods theory of differential equations.
- Integration of more knowledge about physics of a system to capture general physical laws or the specific context of a system.
- Integration and consideration of temporal information.
- Formalised common-sense knowledge and reasoning.
- Dropping postulates, such as non-overlapping intervals, and
- Finding ways to reason about larger, more complex, real-world systems.

All of these points coincide in using all available information for reasoning tasks. Qualitative formalisms are really useful in process supervision when used to establish dependencies between variables taking into account the complete knowledge (physics, experimental knowledge, numeric/qualitative information, dependencies, equations, and so on) related to these variables. Otherwise, qualitative techniques offer only a mechanism for dealing with a poor representation of variables. Consequently, some mechanism for storing and actualising information are needed for these variables. *Object-variables* (previously described in this work) are conceived to assume this task embedding numerical and qualitative information and external parameters supplied by users in the representation of process variables. ALCMEN has been used to perform simple qualitative relationships dealing with these *object-variables* performing simple operations to establish qualitative dependencies between qualitative representation of process variables.

5.4.6. Implementation in MATLAB/Simulink.

The implementation of ALCMEN has been done in MATLAB/Simulink as set of blocks dealing with previous relations and operations. The aim is to add this tool to assist users in developing knowledge-based supervisory systems. ALCMEN static and dynamic relationships have been implemented in MATLAB/Simulink as a set of blocks dealing with qualitative representation of variables under object oriented approach, i.e. *object-variables* (See “Fig. 5.23”). The blocks that form ALCMEN ToolBox, are designed to deal with only qualitative representation of variables, but other blocks can access different fields of objects associated to process variables to carry out numerical operations or to deal with variables history.

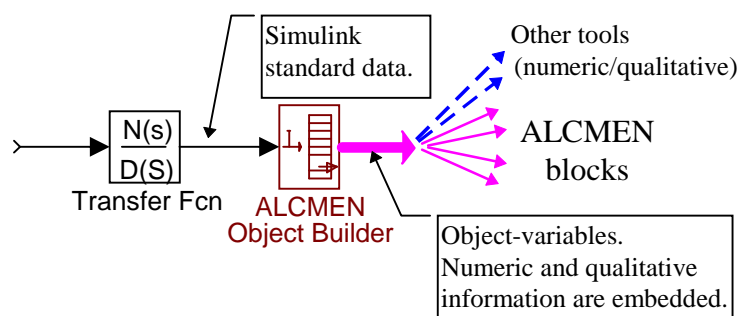


Fig. 5.23 The object-builder allows to deal with significant information. Qualitative representation is accessible to ALCMEN blocks implementing qualitative relations.

This ToolBox has been implemented in order to deal directly with qualitative data stored into *object-variables*. It implies that any ALCMEN block can be

interconnected to the output of an *object-variable* to get the needed information to perform an operation. The complete set of ALCMEN blocks is depicted in Fig. 5.24. The relation with previous defined operators can be extracted from the label under blocks, according to the following notation: the first line describes the associated file and the following lines describe the implemented ALCMEN operator. All of these blocks are implemented to deal with qualitative fields of *object-variables*. Then, both input and output of these blocks are *object-variables*. Although only qualitative attributes are used. The functionality of these blocks is defined according to ALCMEN definitions and *object-variables* structure here after :

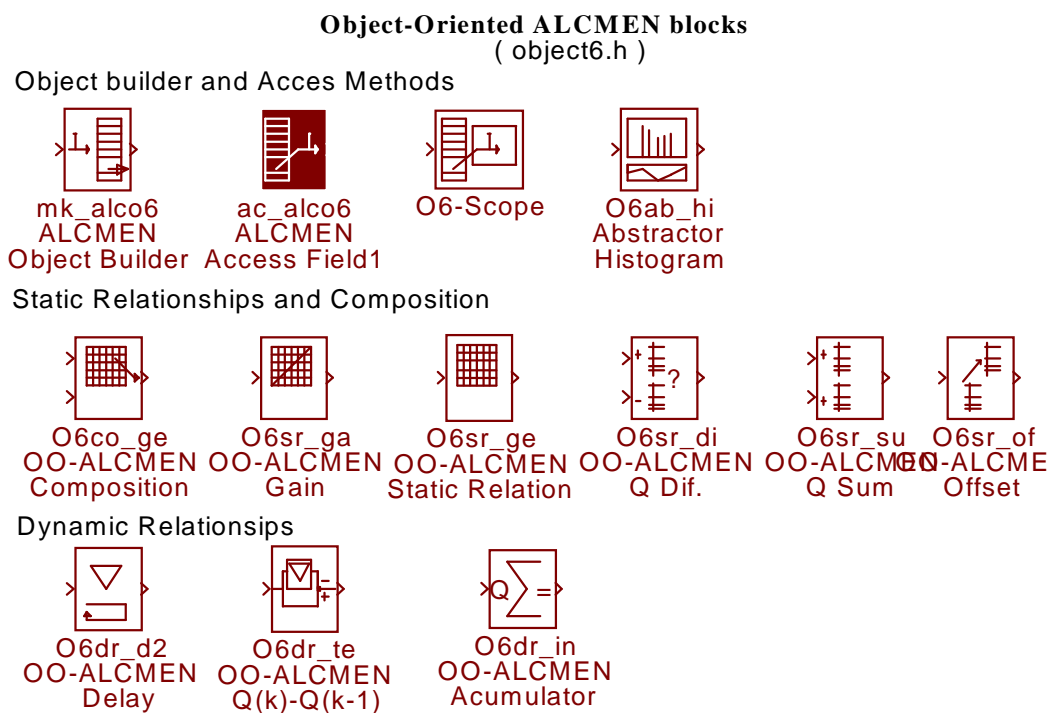


Fig. 5.24 ALCMEN ToolBox for Simulink.

ALCMEN operations represented in Fig. 5.24, are resumed here:

- **Objec_builder**, performs *filtering* and embeds this information in an *object-variable* structure.
- **Access Field**, is the block that allows to access any field of the *object-variable* and supplies this data to the output of the block in standard Simulink format. When the selected field has only a text format representation, this is displayed in the MATLAB command window.
- **O6-Scope**, offers similar functionality to the previous block but the information is presented graphically by means of a scope. The text attributes as description or units, are printed in the MATLAB command window.

- **Abstractor Histogram** is an *object_builder* that performs numerical to qualitative interface using the histogram abstraction tool for this purpose. Several indices are available.
- **Q_dif**, performs a difference between qualitative indices of two input *object-variables*.
- **Q_Sum**, performs the addition between qualitative indices of two input *object-variables*.
- **Composition**, allows to perform any relationship between qualitative variables that can be represented by a table. Indices of input are used to fire the output from the table.
- **Gain** is the block that performs a product of a qualitative variable and a constant. The output index is selected from the set of input indices.
- **Static relationship block** is used to implement any variable function. A table is used to define the input/output dependency. This could be used to implement extremal functions.
- **Offset block** allows to shift indices of input variable. It is similar to an addition, but the result is not filtered with input indices.
- **Delay** stores the whole input *object-variable* during a sample time to delay the output.
- **Tendency** of a qualitative variable is obtained by qualitative difference of an input and its delayed value.
- **Accumulator**, is conceived to perform the integration of qualitative indices of the input variable. The user can select the integration rate.

Additional blocks have been added to ALCMEN relationships, in this ToolBox, for building *object-variables* from habitual numerical values supplied by Simulink and to supply data from desired field of *object-variables* to standard Simulink blocks (“Fig. 5.23”). Thus, a single line is used to represent one *object-variable* and all features, numerical and qualitative, are embedded and actualised in the same structure. The restriction is that all blocks interconnected to this line must know how to access the desired fields.

5.5. Expert System CEES into MATLAB/Simulink.

CEES, [De la Rosa J.Ll, 1994], (C++ Embedded Expert System Shell) was firstly conceived for supporting co-operative ES. From that time on, CEES was subject to many modifications and extensions in order to make it useful in industrial process supervision. Several efforts have been done in the context of this thesis; see for example [Sàbat, 1996] and [Martinez, 1997]. This effort has resulted in a new version of this shell, CEES 2.0. It has been developed in the scope of this thesis following two parallel and complementary lines. The first one

was to achieve a stand-alone application with external communication capabilities and graphical user interface for assisting the definition of knowledge through the development of production rules. This version has been called LabCEES. The second, and more interesting from the point of view of this thesis, consists in embedding the CEES 2.0 inference engine into a framework with other tools for assisting the design of supervisory systems. It is called SimCEES. With the same goal as previous presented tools, ALCMEN and *abstraction tools*, SimCEES has been embedded in MATLAB/Simulink to assist supervisory systems design in a friendly to use framework (CASSD). The use of a graphical block representation for an ES representation facilitates the comprehension and use of this tool. The goal is to use the stand-alone application, LabCEES, for the final application and the CASSD framework in the developing time. The benefit of this redundant implementation of the ES is that rule bases could be directly used by either.

5.5.1. CEES features. Advantages and drawbacks.

Object-oriented approach/methodology constitutes the core of this shell (CEES) because it is intended to exhibit capabilities of encapsulation of information, polymorphism of methods and data abstraction. Four basic classes were defined to deal with *variables*, *facts*, *inference engines* and *simulators*. Deductions are performed by *inference engines* through reasoning about its input *facts*, but they also have the capability of asking for information to other *inference engines* (*facts*) or *simulators* (*variables*). In the new version (LabCEES and SimCEES) some of these classes have disappeared and only remains the classes *inference-engine* and *facts*. CEES main capabilities are pointed here :

- Facts are the kind of data supplied to the ES to perform some reasoning. Facts are objects embedding both, numerical and/or qualitative data and subjective information about this. For example certainty is used to represent doubt about source of information and fuzzy sets can be build around numerical data to perform fuzzy comparisons.
- CEES is a rule-based system with specific syntax and forward chaining inference mechanism.
- Co-operation among ES is performed by exchanging information (encapsulated as objects representing *facts*) between *inference engines* and asking *simulators* about *numeric* variables.
- Qualitative values are defined in terms of orders of magnitude. Simple operators were added to deal with them. Methods and qualitative fields are encapsulated in the same object associated to numeric *variables*.

- Reasoning with temporal windows. Data history is stored in CEES *variables* in temporal sliding windows for making inference about trends of process variables in the temporal window.
- Fuzzy reasoning is incorporated in CEES on the base of logical comparison of *variables* by means of the *equal*, *lower* and *greater* operators.

Mainly, due to the fact that CEES is based on C++ objects, some advantages turn into shortcomings at the moment of developing the ES application. Some of these points are highlighted below:

The main advantages are :

- CEES is a flexible open shell subject to continuous evolution. Addition of methods or objects to solve specific situations is allowed.
- CEES variables are conceived as objects with encapsulation of data and methods similarly to the *object-variable* structure described in the previous sections.
- The implementation of *facts* used in CEES is directly applied as a higher level data structure of *object-variables*. Thus, expert reasoning about *object-variables* is accomplished directly.

Some drawbacks can also be pointed:

- The current version of CEES is not user-friendly. The user must be familiarised with C/C++ programming and a training period is needed before starting the development of ES applications.
- The use of C++ objects, with complex structure, in the definition of systems (*simulator* and *inference engines*) and variables (*variables* and *facts*) makes the interpretation of global architecture of ES difficult.
- No external interface is provided with CEES. Despite of this drawback, DDE- Dynamic Data Exchange and any C/C++ library can be directly used.

New version, CEES 2.0 has been developed in this thesis to avoid this drawbacks. This is CEES 2.0 and the implementation has been designed for two platforms : LabCEES, directly under Windows and SimCEES under Simulink. CEES has evolved towards two separated and complementary products (LabCEES and SimCEES), especially conceived to be used by process engineers :

- LabCEES is a stand-alone shell running under Windows. A friendly user interface has been added to assist in configuring ESs architecture. This is, the number of inference engines, rule bases, facts bases and related parameters. A graphical user interface assist user in these tasks.

- SimCEES, is conceived to run under Simulink as a block. A SimCEES ESs are conceived to deal with *facts*. Thus, both, input and output are facts. Output is obtained as deduced facts from reasoning about input facts. Any ES block is defined according to the structure of a rule-based knowledge. Multiple architectures could be defined by adding or moving ES blocks.

An additional feature of CEES 2.0 is the exportability of rule bases between SimCEES and LabCEES. This increases the flexibility to test and compare different structures and knowledge bases before the final implementation. Modularity is present in both implementations.

5.5.2. SimCEES : Implementation of an ES in MATLAB/Simulink.

Simulink has been chosen to develop a set of ToolBoxes to deal with the design of expert knowledge Supervisory systems. In previous sections *abstraction tools* have been presented to process numerical information and to interface with qualitative representation of process variables. ALCMEN has been introduced as a mechanism for representing simple relationships between qualitative data. When more complex knowledge structures are needed to represent special expertise, as is the case of expert supervision applications, then ESs are used. Therefore, the implementation of an ES Toolbox for Simulink is needed to complete the framework for supervisory systems design.

The shell CEES has been chosen to be integrated into Simulink as a new block. As a result, this shell has evolved to SimCEES (CEES 2.0) especially conceived to work under the block representation. The object structure of CEES has been simplified for this purpose. The *simulator* class has been suppressed because this facility is supported by Simulink (for numerical simulation) and ALCMEN (for qualitative simulation). The class *variable* has evolved to *object-variable* definition to preserve the integration mechanism. Then, only *facts* and *inference engines* remain as SimCEES classes. *Facts* are the objects supplied to the ES at its input and deduced and available at the output. *Inference engine* include the reasoning mechanism used for these ES. *Facts* are basically objects encapsulating *object-variables* and other attributes (*certainty* and *landmark*, for instance) and methods needed for the *inference engine* mechanism.

The Simulink implementation of SimCEES is represented in Fig. 5.25. Any Simulink block used to represent a knowledge base is formed by an inference

engine and a rule-based. In execution time, facts base is created and actualised according to deductions performed.

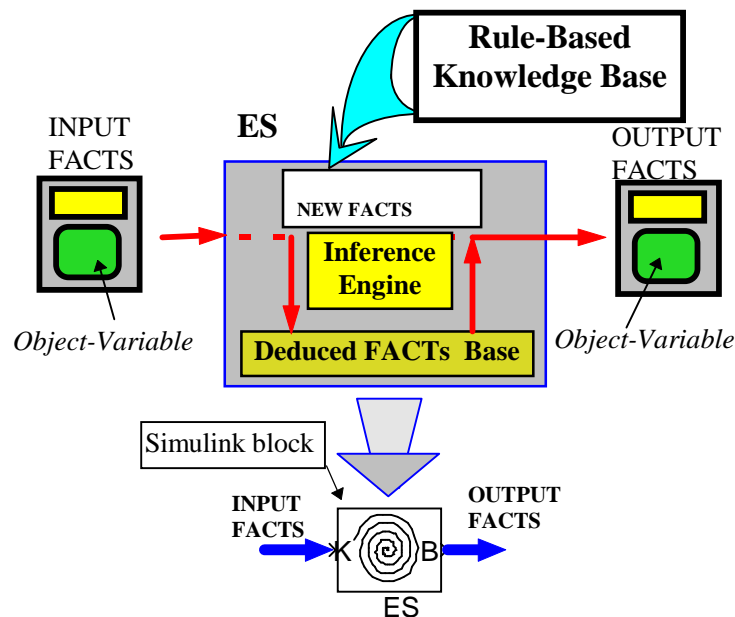


Fig. 5.25 Simulink block implementation of an ES

SimCEES is intended to work on-line with process variables (*object-variables*), encapsulated as input *facts*. Consequently, it works as a discrete system according to a sampling period. The input *facts* encapsulate the information related to process variables, *object-variables* is actualised each sampling time. Consequently the rules base must be evaluated, by the inference engine, at any actualisation of input data to deduce new *facts*. Results are also given as *facts* and the outputs are selected from these deductions. The benefit of working with *facts* only, at input and output, gives the possibility of connecting directly to another ES blocks at the output, as depicted in Fig. 5.26.

The main benefit of modular block representation, supported by Simulink, when dealing with ES is the distributed knowledge representation because of:

- Modularity : The partition of large KB into smaller KBs.
- Specialisation and reusability of knowledge bases. Supervisory systems should be designed to reuse knowledge, especially in distributed AI systems where the same KB can be applied in solving similar problems in the same process. For example, expert tuning of PID is a domain where the same KB can be reused for several purposes. These KB can be reinforced by process knowledge declared in other KB although basic rules are preserved.

- Flexibility, in designing and testing ES-based architectures. Modifications in the architecture of the whole ES can be performed by simply moving a block.
- The implementation and easy testing and comparison of special architectures based on redundancy, competition, reinforcement or co-operation.

Some problems are inherent to the choice of using Simulink interface instead of developing a new one for SimCEES. Despite the graphical support for manipulating ES as a single block used in Simulink for representing an ES, there is no support for editing rules. The user must use a standard editor for this purpose.

5.5.3. Expert Reasoning with SimCEES.

In the previous section the benefits of modularity of SimCEES have been introduced. The graphical support of Simulink by means of block representation permits an easy reconfiguration of the ES associating a block to each KB. When working with large and complex problems, the solution is easier if it is divided into sub-problems, because small knowledge bases are better structured. Modularity in SimCEES is possible because a single structure for data representation is used. All available information in *object-variables* (numeric, qualitative, symbolic or logic) coming from process variables is embedded together with attributes and methods needed for the ES inference engine into a new object structure called *facts*. These *facts* preserve the same structure for any ES input and output. Consequently, inference engines must reason about evolution of input *facts*, firing the appropriate rules in the knowledge base to deduce about these input *facts* or other previously defined in the knowledge base. ES outputs are selected from deduced *facts* base at each sampling time.

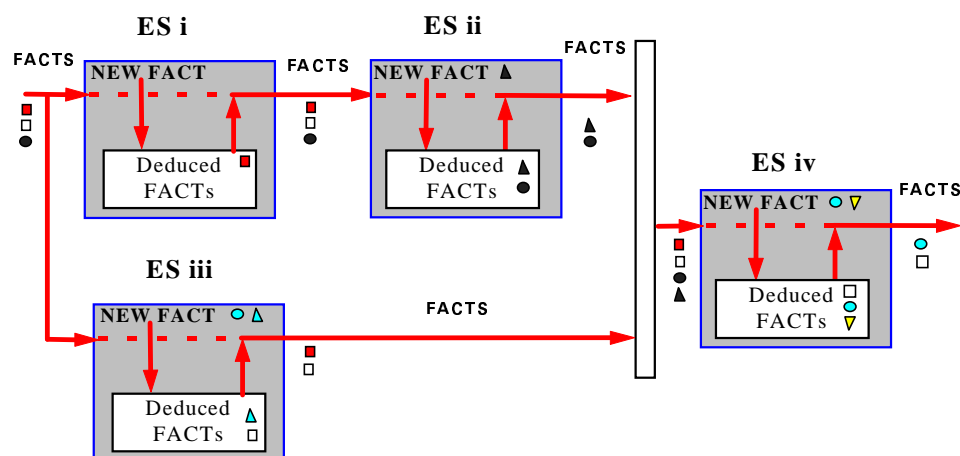


Fig. 5.26 Input and Output of ES blocks are facts.

In practice, output can provide not only deduced *facts*, but also input *facts* to facilitate the flow of *facts* among ES blocks, avoiding an unnecessary amount of lines connecting blocks. *Facts* declared in a knowledge base cannot be supplied as output, except when they have been deduced. When using parallel configurations of ESs blocks (the output of these ESs, see ESii and ESiii in Fig. 5.26, are supplied to the same ES, ESiv in Fig. 5.26) there exists the possibility that the same *fact* will be supplied, at the same sample time by two or more different blocks as input of another ES block. In such cases there exists a possibility to differentiate those *facts* or use only one of them. In case of the second option is chosen, priority is given by graphical connection order.

An example of how this mechanism works with *facts* is depicted in Fig. 5.26. In this figure, *facts* are represented by symbols as ■, ●, ●, ▼, ▲, □, ● and ▲. The evolution of these facts is represented in a sampling time, showing the *facts* supplied for four interconnected ES blocks. For instance, ESi deduces ■, while at the same time it can also supply input *facts* to be processed by the next block. On the other hand, ESii can only supply deduced *facts*. It can be observed that some deductions are from external input, ●, while other are defined in the same ES, ▲. A different situation is presented in ESiv. In this case, not all deduced *facts* are supplied at the output (See *fact* ▼). A reason for this, could be that this deduction causes a message to be printed; therefore, it is not needed at the output.

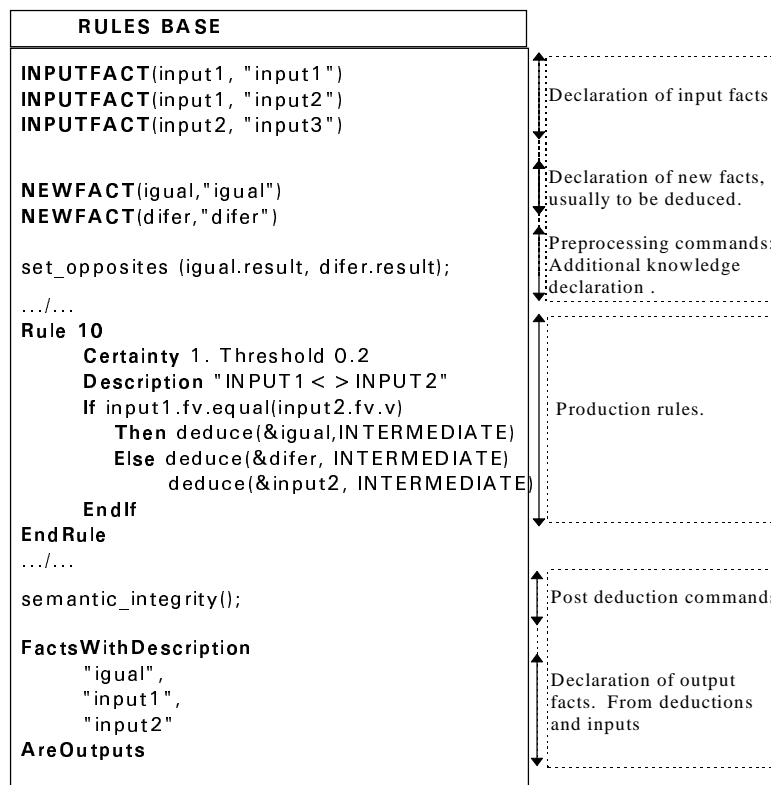


Fig. 5.27 Example of rules based knowledge declaration in SimCEES.

A simplified example of how rules are described in SimCEES and declaration of input and output *facts* is shown in Fig. 5.27. This could be the rule base associated to ESiii, where output *facts* are selected from input (■-“input1”, □-“input2”, ●-“input3”) and deduced (▲-“difer” and □ -“input2”). *Facts* are used at input and output to provide a *fact* flow. This is needed when working with dynamic systems and *facts* represent some abstracted feature of this behaviour. The final goal is to use the *facts* to close the supervisory loop.

5.5.4. Exportability of rules. LabCEES, the stand-alone application.

The new version of CEES (CEES 2.0) has a double implementation. The first, SimCEES, is described in the previous subsection and runs under Simulink. The second, LabCEES, is complementary to the first one, and is conceived as a Windows based stand-alone application. LabCEES is still under development, but the goal is to provide this shell with a graphical user interface for assisting ES configuration and rules production. LabCEES is an open system with communication capabilities for interfacing the application with monitoring systems and other Windows based programmes using dynamic data exchange (DDE).

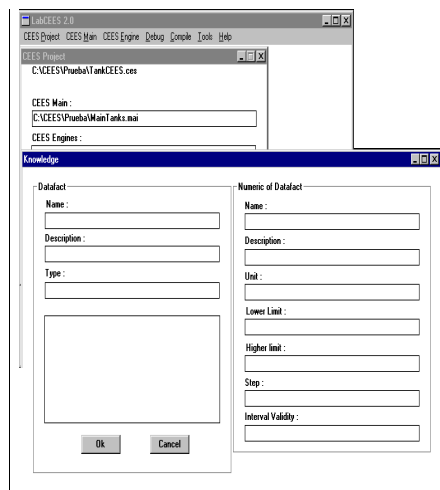


Fig. 5.28 LabCEES windows based graphical user interface.

LabCEES is prepared for dealing with modular ES, structured in more than one inference engines. To achieve this goal, *inference engines* are provided with communication mechanisms to interchange deduced information (*facts*) according to their own rule base. This approach permits to reuse rule base in SimCEES, building identical structures representing the inference engine as independent Simulink blocks.

5.6. The global framework.

Object-variables, embedding data and methods, implementing numeric to qualitative interfaces, based on *abstraction tools*, for dealing with ALCMEN qualitative relations and CEES expert rules, have been implemented by means of Simulink block-based graphical representation. As a result, MATLAB/Simulink, the numeric CACSD framework, has been improved to support expert supervisory strategies design becoming a CASSD framework.

5.6.1. Architecture and tools.

The previous described tools, *abstractors*, ALCMEN and SimCEES, have been implemented in Simulink under an object oriented approach to facilitate the task of designing and testing supervisory structures, based on expert knowledge [Melendez et al., 1996b][Melendez et al., 1997a]. *Object-variables* have been defined to encapsulate information from process variables at different levels of abstraction. Then, *abstractors* also called *abstraction tools*, are encapsulated in *object-variables* to obtain a significant information from those variables related to the process. This information is encapsulated using both numerical and qualitative representations in the *object-variables*. This data can be directly supplied to a modular ES, SimCEES, by building *facts* that join all the information encapsulated in *object-variables* and subjective appreciation about certainty of this information with methods needed for the inference engine of ES. Parameters are supplied using a dialogue window.

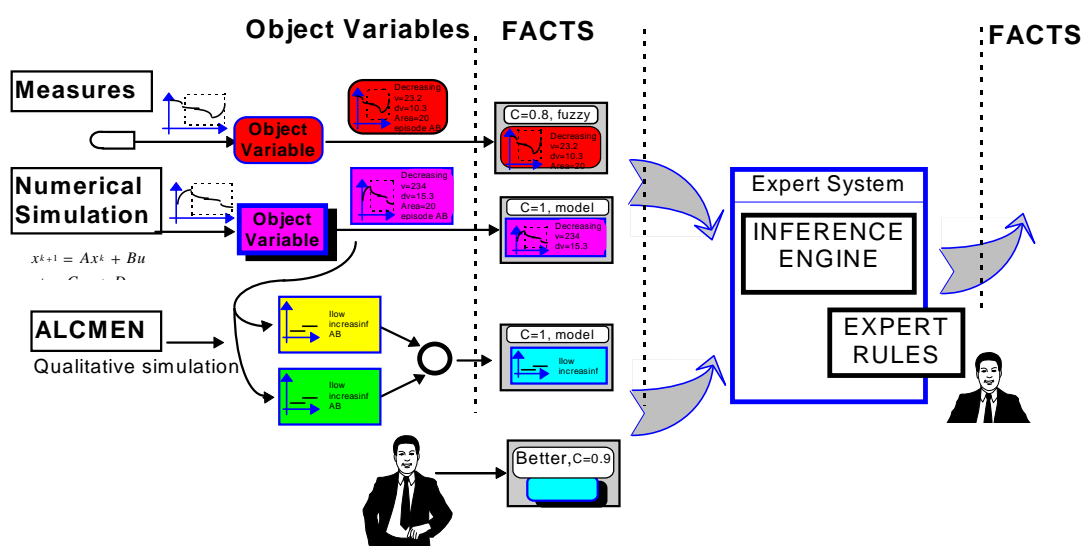


Fig. 5.29 Mechanisms for obtaining significant information involved in expert reasoning.

Because not all expert knowledge concerns numerical information, ALCMEN is added to deal with qualitative information, allowing the designer to implement simple relations in order to obtain a rough evolution of not measurable process variables. In the implementation of ALCMEN blocks, *object-variables* structure have been used, despite of only qualitative fields are used and *abstraction tools* are only used in the interface blocks (*filtering*), to preserve the structure and encapsulation of information used with numerical data.

Fig. 5.29 shows how those tools are involved in the procedure of obtaining significant information to be supplied to the ES. The heterogeneous kind of information merged to reason about complex systems, needs the object-oriented approach to be used as integration mechanism. In this case, it has been used in the representation of information flow (object-variables and facts) and also to build a modular ES able to deal with this encapsulated information.

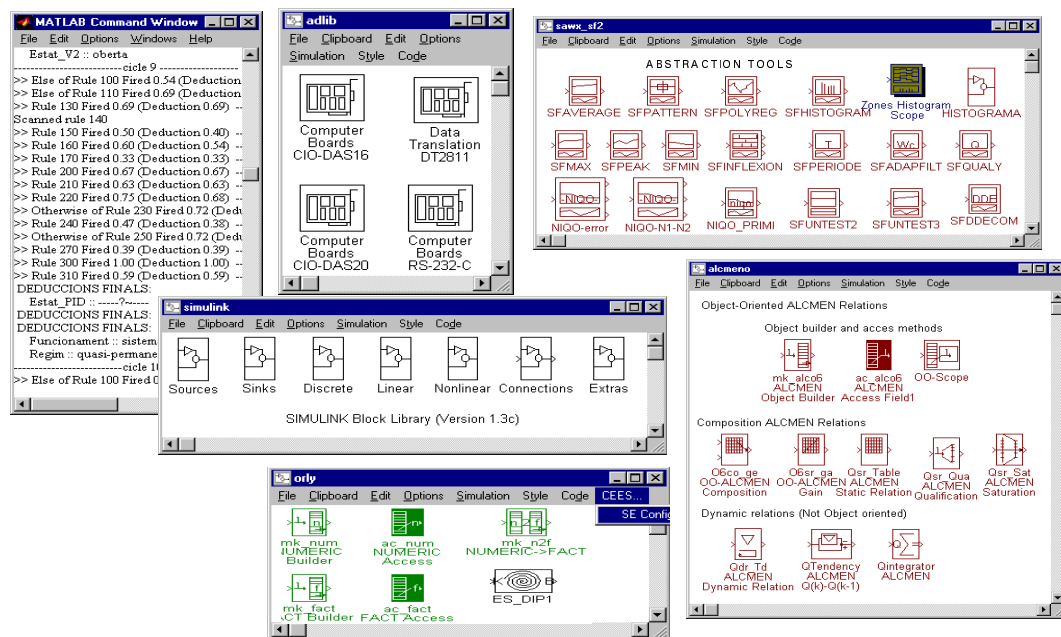


Fig. 5.30 Set of Simulink ToolBoxes of the proposed CASSD framework. Abstraction tools, ALCMEN and CEES are available.

The previous schema is implemented as sets of blocks grouped in ToolBoxes for each tool (CEES, ALCMEN and Abstraction Tools) forming the complete CASSD framework.. The use of this CASSD framework for designing supervisory structures allows partial validations of the structure when working with complex systems. Simulink capabilities allow to group set of blocks into a new one simplifying the appearance and allowing knowledge encapsulation. At the same time connectivity between tools is quickly solved by tracing lines between inputs and outputs of blocks.

5.6.2. Knowledge representation.

Processing and administration of expert knowledge requires of knowledge formalising and structuring. Rule-based programming is not the only technique used with this goal, within artificially intelligent programs, but nowadays, it is the more adequate technique for us to directly codify expertise. In spite of this, some other classical formalism are more or less present in the proposed framework taking in account the particular point of view of process supervision domain. Procedures typically used to represent objects and their relationship are used to represent process variables :

- *Production rules*, for describing knowledge in the form of “IF...THEN...” rules are present in CEES. Knowledge base associated to any ES block is described as sets of IF-THE rules, as showing Fig. 5.31.

<pre> Rule 10 Certainty 1 Description "Regim status: Transient " IfOr tendency_level->fv.v == great Or tendency_error->fv.v == great Then deduce (INTERMEDIATE, transient) EndIfOr EndRule </pre>	<pre> Rule 210 Certainty 1. Threshold 0.45 TraceHere Yes Description " Valve 1 Closed . " IfAny (PF) PF->result == anomalia2 And Alternative levels_dif->ups_in_interval() Or levels_dif -> fv.greater(dnivells->fv.low) EndAlternative Then deduce(INTERMEDIATE, v1_closed) Otherwise deduce(INTERMEDIATE, v1_open) EndIfAny EndRule </pre>
---	--

Fig. 5.31 SimCEES rules

- *Semantic networks* : Graphic representation of knowledge based on objects and their relationship can be compared with the block-based graphical distribution of knowledge provided by the graphical user interface of Simulink. Interconnection of blocks represent dependencies between process variables and the flow of information. This is the case of ALCMEN relations or dependencies between several ES blocks.
- *Frames*, are thought to be data structures for representing objects. In this work the object-oriented approach has been used for representing *process variables* and information related to process behaviour. All of these information is embedded into *object-variables*.

The modular conception of all those tools presented to help supervisory systems design, supported by Simulink blocks, simplifies modular design of

reasoning systems. Simplicity and modularity, which combines quantitative and qualitative knowledge, are important advantages when building on-line reasoning systems that are embedded in conventional applications. According to the degree of abstraction of knowledge about process variables the tools presented in the previous sections are especially conceived to assist in three hierarchical levels (Fig. 5.32):

- *Level of Signal* : This is the interface level between the supervisory structure and the process. It is the acquisition level where measures from process and process engineers knowledge join together to supply other levels with significant information extracted from measures. The tools used to obtain qualitative information from measured signals are the *abstraction tools* or *abstractors* and their output are a qualitative representation of the trends of process variables (tendencies, oscillation degrees, alarms, degree of transient states, ...) needed not only in supervision, but also in fault detection and diagnosis tasks. The use of *object-variables* for this purpose is similar to a representation based on *frames*.
- *Level of Qualitative Relations* : At this level is where engineers could represent dependencies or relations between qualitative variables, obtained or not from the first level. These relations could be used to deduce roughly inaccessible dynamics according to a qualitative model, physics or just observed dependencies between variables that are difficult to join in a numeric equation. ALCMEN is used to perform these tasks (and knowledge is represented by *semantic networks*).
- *Level of Rules* : This level reflects dependencies between facts that could be represented by rules describing expert KBs. Thus, ES are used at this level. This is the final destination of inferior levels.

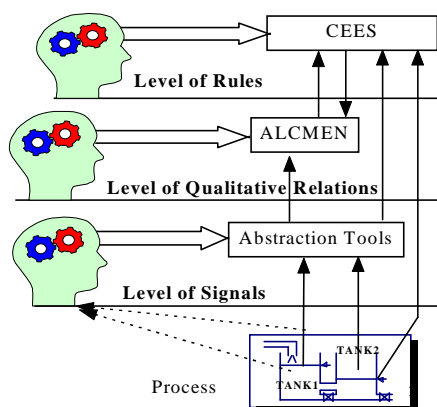


Fig. 5.32 Levels of abstraction

Each level is able to deal with significant information coming from equal and inferior levels (See "Fig. 5.32") . The mechanisms used in the representation of

knowledge at each level are associated with the respective tools, described in previous subsections. These mechanisms, are always implemented using object-oriented programming and encapsulated as blocks in a graphical representation supported by the Simulink graphical user interface. Knowledge modularisation and encapsulation is represented in Fig. 5.33.

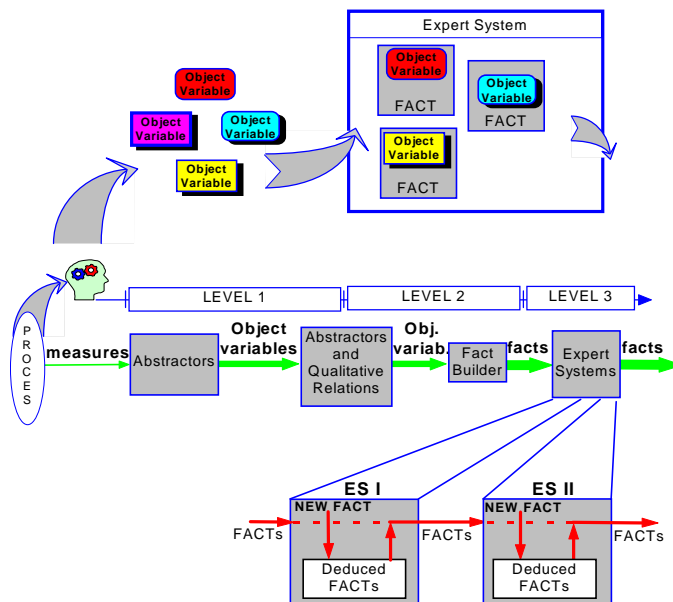


Fig. 5.33 Encapsulation of knowledge in *object-variables* and *facts* related to the tools used with this purpose in each level.

Raw numerical data coming from process (measures or estimations) are encapsulated into *object-variables* together with abstraction tools used to obtain qualitative descriptions of process variables based on significant information. Qualitative description of those signals are stored in temporal windows to be used in reasoning task by other blocks. Finally they, can be supplied to ES as *facts* encapsulating additional information.

5.7. Conclusions.

A modular implementation of tools for dealing with expert knowledge and avoiding interfacing problems has been presented. The implementation of such tools has been done under MATLAB/Simulink. The object-oriented approach has been used in the variables implementation, to embed both data and abstraction methods, and in the design of several tools to deal with information at different abstraction degrees. Abstraction tools have been presented as numerical to qualitative interfaces to provide simple qualitative representation, event generation or temporal episodes. Qualitative relations are thought to be managed

by means of ALCMEN representation language, using block representation to perform simple operations with qualitative representation of signals. Both static and dynamic qualitative relationships have been implemented. An ES shell CEES has also been embedded into a Simulink block, encapsulating expert rules to reason about input *facts*. This shell provides a forward chaining inference engine and fuzzy reasoning for imprecision and uncertainty management.

The use of object oriented technologies is necessary to integrate such tools in a framework for assisting expert control and supervision. This methodology could be useful for large projects because of the modularity and encapsulation of knowledge into blocks.

6.

Some illustrative examples in using the CASSD framework

6.1. Introduction

In the following subsections some developments carried out in the proposed framework are described to demonstrate the capabilities of the integrated tools and the benefit of avoiding data type conversion. Integration pitfall, derived from the necessity of multiple representations of information related to process variables, has been overcome. The advantage of disposing of numerical and qualitative methods for dealing with process variables is tested. Fault diagnosis, qualitative observers and estimations are exemplified by means of simple problems derived from real process. Although, these examples are an intermediate step forward to complete supervisory schemes, they are enough to show the benefit of disposing of a framework to assist such developments. Different situations have been proposed to illustrate the general usability of this

proposal, avoiding particular point of view derived from a single problem discussion. Consequently, special interest, in the explanations in the following examples, is focused on describing framework capabilities more than in discussing proposed solution.

6.2. Fault diagnosis of a simulated plant.

The tools described in the previous chapter have been tested in a laboratory plant. A model of this plant has been obtained for testing fault detection structures and the flexibility to modify these tools. The goal is to use the model for developing a knowledge-based fault detection structure and then to use the same CASSD framework to test and validate the design by means of substituting the model by specific blocks to access hardware (plug in boards). The example presented in following paragraph was firstly developed using different applications ([Melendez et al. 1995]), externally linked, for solving data management (MATLAB/ Simulink for signal processing and simulation, G2 as rule-based ES). An additional application was developed to manage and link the environment. The application has been rebuilt using the CASSD framework, (See [Melendez, et al., 1997b]) as it is explained in the text.

6.2.1. Laboratory plant description.

The laboratory plant is composed by two coupled tanks connected by two pipes, as depicted in “Fig. 6.1”. The liquid of tank_1 is spilled into tank_2 through two pipes.

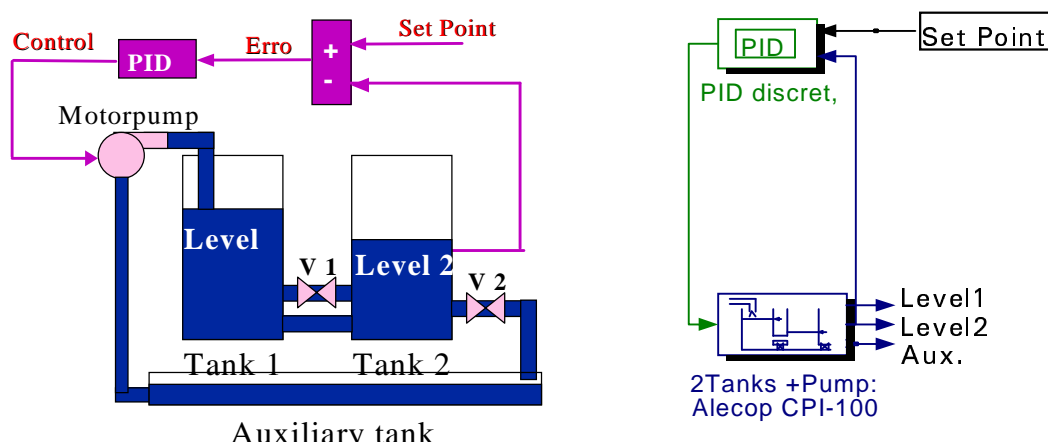


Fig. 6.1 Laboratory plant: Two coupled tanks.
Simulink model.

The flow between both tanks could be modified by closing a valve, placed in one of the interconnecting pipes (V1). Tank_2 spills liquid, depending on the state of the output valve (V2), to an auxiliary tank placed below of the laboratory plant. This auxiliary tank is used as fluid source to pump liquid to the tank_1. A PID measuring the level of tank_2 drives a motor-pump filling tank_1 in order to reach the desired set point of level 2. The magnitudes of this system are restricted to 31 cm. (tanks height), for the measure of both levels, 0-10 volts for the control signal. The evolution for several changes in the set point is represented in Fig. 6.2.

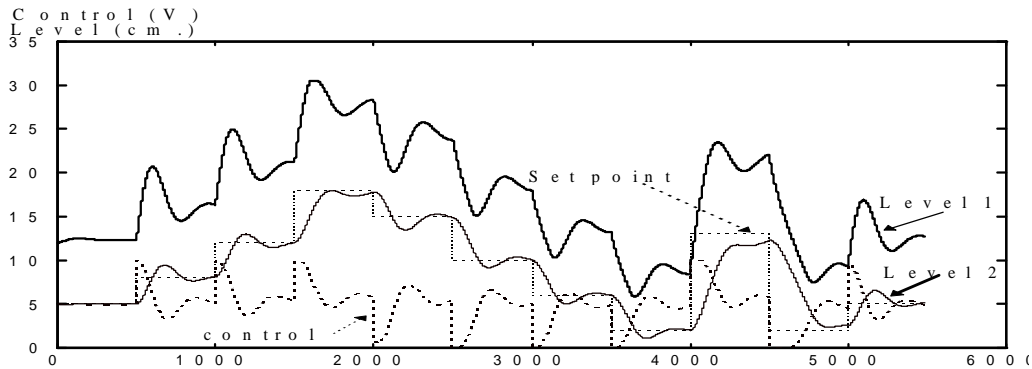


Fig. 6.2 Evolution of plant variables to several changes in the set point.

The normal operation mode is defined when both valves (V1 and V2) are open and the PID parameters are accurately tuned to regulate level 2. The pump is also supposed to work correctly. Fault situations have been reproduced by closing valves (to simulate an obstruction of pipes) and switching the pump off. Levels of both tanks and the set point are the only available measures. Four possible malfunction situations to detect are typified. Simultaneous situations are not considered to simplify the example.

Situation 1	Situation 2	Situation 3	Situation 4
Valve 1 closed	Valve 2 closed	Pump crashed	Bad regulation

Table 6-1 : Four situations have been defined.

When failures are introduced, then the expert diagnostic system must be able to detect and identify them. Therefore, the goal of the supervisory system is to track the process and detect situations that incite failures or process malfunctions, as well as to know when the process works in the normal operating conditions. With this goal, existent numerical tools and representation capabilities have been used to study the influence of faults in the behaviour of the process.

6.2.2. Implementation of a knowledge-based fault detection system in the CASSD framework.

In order to test the CASSD framework to design a fault detection system, two representative signals have been studied : the *difference of levels* (level tank_2 minus level tank_1) and the *error* (set point minus level tank_2). These two signals have been used because of process behaviour is reflected on them. The *error* signal joins the evolution of the set point and the dynamics of the level of tank_2. The *difference of levels* supplies information related to both tanks and the interconnection between them. The evolution of these signals has been studied in several simulations of normal operation conditions and fault situations as described in last paragraph (Fig. 6.3 and Fig. 6.4). Although, numerical signals are provided from simulation, human reasoning is performed at a more abstracted level of these signals. Visual appreciation of signals evolution is used to define process behaviour. Consequently the set of rules obtained for supervising this process uses a qualitative interpretation of numerical signals.

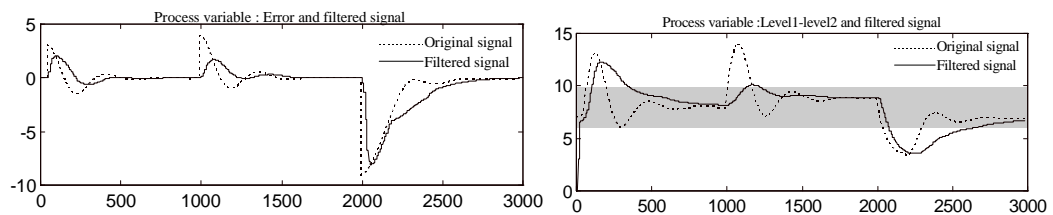


Fig. 6.3 Evolution of abstracted data when several set point changes are performed.

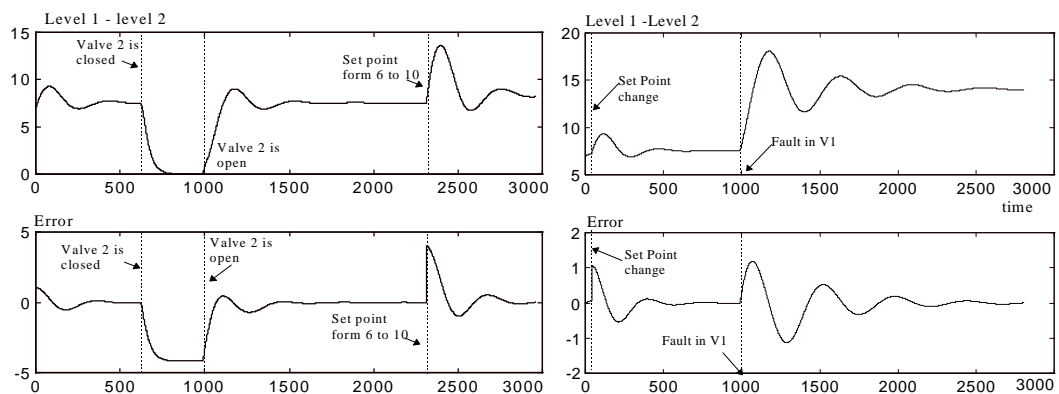


Fig. 6.4 Process dynamics is reflected in the evolution of error and difference of levels. The figures represents when valves are closed and changes in the set point.

It means that situations are described by experts from trends of signals or deviations from normal situation and global appreciation of the regime of the

process (transient state or steady state). For instance, in the normal operation mode (when the valves are not closed, PID is well tuned and motor-pump is running) difference of levels, in permanent regime, remain in a short range between 6 and 9.5 cm (See Fig. 6.3). Another observation example is shown in Fig. 6.4, when valve 1 is closed, then the system is stabilised in a longer time than in normal operation conditions. This appreciation about steady state are easier to represent in rules than those related to transient state. Because acquired knowledge is not only related to numerical values of signals, it is necessary to interface both numerical signals (from simulated process in this case) and KB defined according to qualitative appreciation of these signals. With this aim qualitative representations of some features of numerical signals will be useful. This task is performed by the *abstraction tools*. *Abstractors* are defined to supply ES with qualitative and numeric significant information obtained from numerical signals. In the example qualitative tendency and qualitative deviation from this have been used together with a smoothed signal of original process signals (See Fig. 6.5).

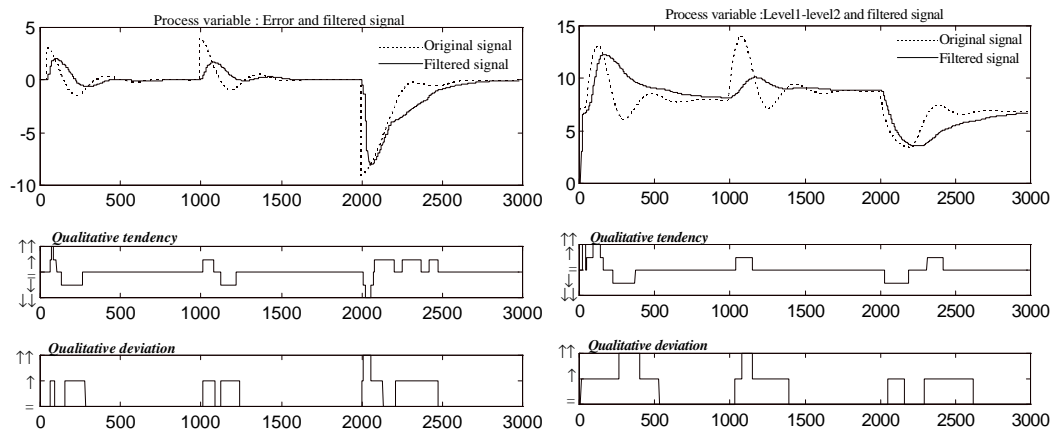


Fig. 6.5 Evolution of abstracted information (qualitative tendency and deviation) when several set point changes are performed.

```

If (error tendency is =)
  And (error deviation is =)
  And Alternative (level1-level2 deviation is ↑↑)
    Or Not (level1-level2 tendency is =)
    Or (level1-level2 secondderivative >= 0.15)
  EndAlternative
Then deduce transient
EndIf

```

Facts	Attributes
error	<i>tendency, deviation</i>
level1-level2	<i>tendency, deviation</i>
transient	<i>secondderivative</i>

Fig. 6.6 Expert rule, in pseudo code, reasoning with numerical and qualitative information for deducing about transient state.

Using qualitative representations of process variables, it is easier to supply adequate information about process to rule-based ES to diagnose about behaviour. For example a rule deciding about transient state could be defined according to both numerical and qualitative descriptions of information related to process variables (Fig. 6.6).

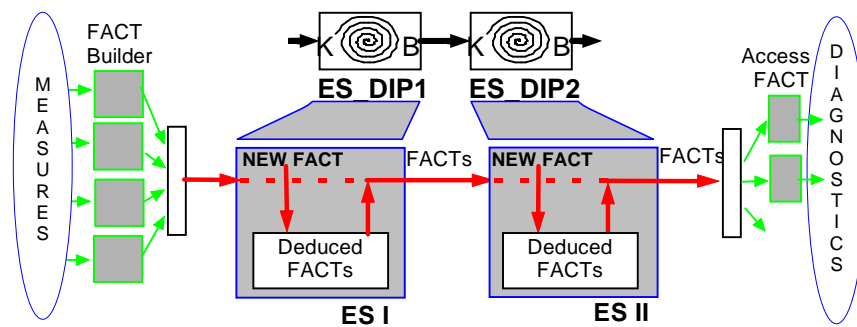



Fig. 6.7 Modular implementation of ES in two levels. Output facts come from deductions or directly from the input.

Fig. 6.7 show the how the ES deals with input data supplied as *facts*. The first ES block decides about system regime (transient, permanent) and deduces simple situations of abnormal operation mode assigning a certainty value to these new *facts*. The second level yields final deductions about process behaviour when a fault is provoked. This second level uses both, deductions of previous level (ES block) and input *facts* supplied to first block. The division in two levels is performed after observing the KB obtained from observations and structuring it as two sets of rules.

Final implementation of the complete system in the proposed CASSD framework, is depicted in Fig. 6.8. The information elaborated by the *abstraction tools* is supplied to the ES for fault detection. *Facts* are the kind of data structure that the ES can manage, then a dedicated block, , performs this task (See Fig. 6.8). In this case, input *facts* are obtained from : *set point*, *difference of levels* (qualitative tendency, qualitative deviation and filtered signal), *error* (qualitative tendency, qualitative deviation and filtered signal), and *time*. The ES is structured in two KB for reasoning about process behaviour using abstracted information. NIQO error and NIQO N1-N2 represent the abstraction tools applied to signals error and difference of levels to supply qualitative information about its qualitative *tendency* and *degree of deviation* respect this tendency. Numerical input is filtered and also supplied to the ES. This information is encapsulated as *facts* to be used by the ES. This is the task of blocks connected as input to the multiplexer (Mux) in Fig. 6.8.

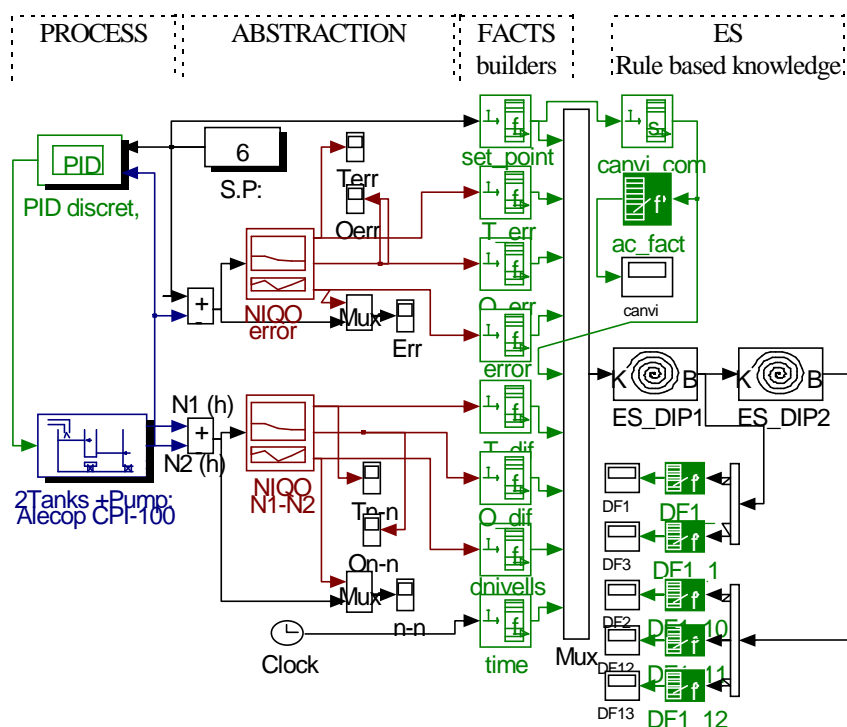


Fig. 6.8 Implementation of an ES based diagnostic structure in the CASSD framework.

Abstraction tool used in this example has been implemented by means of Simulink blocks. Thus, qualitative information provided by NIQO block is obtained smoothing input data for rejecting oscillations. Then, a qualitative tendency is obtained from derivative of this smoothed signal splitting the amplitude space in five zones (five categories or labels of tendency are obtained). At same time smoothed signal is subtracted from original signal for obtaining a qualitative index related with the deviation respect to tendency (in this case only three possible qualitative deviations are used). A complete description of this abstractor can be consulted in [Colomer et al. 1996] and Fig. 6.9 for a graphical representation ins Simulink blocks. Thus, each NIQO block supplies three output (smoothed signal, qualitative tendency and qualitative deviation respect this tendency).

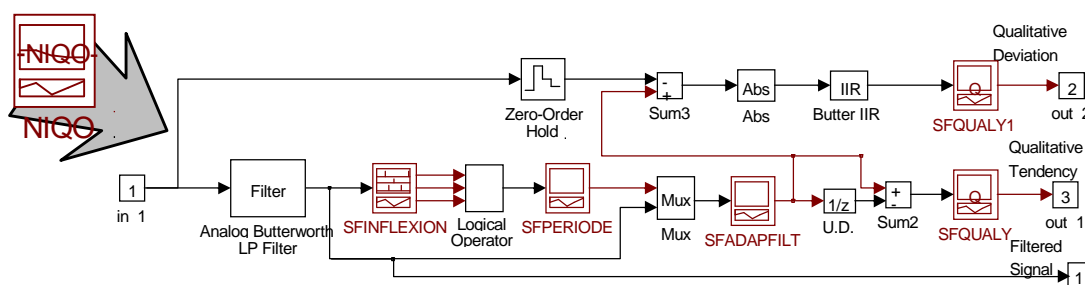


Fig. 6.9 Abstraction tool implemented by means of Simulink blocks.

Evolution of difference of levels is represented in Fig. 6.10 when "Valve 1" is closed to simulate and obstruction in the interconnecting pipe. The evolution of deduced diagnostics is reproduced below abstracted information. Diagnostics are given in a permissible delay of time. The elapsed time for diagnostics could be reduced but this implies that uncertainty of these deductions increases.

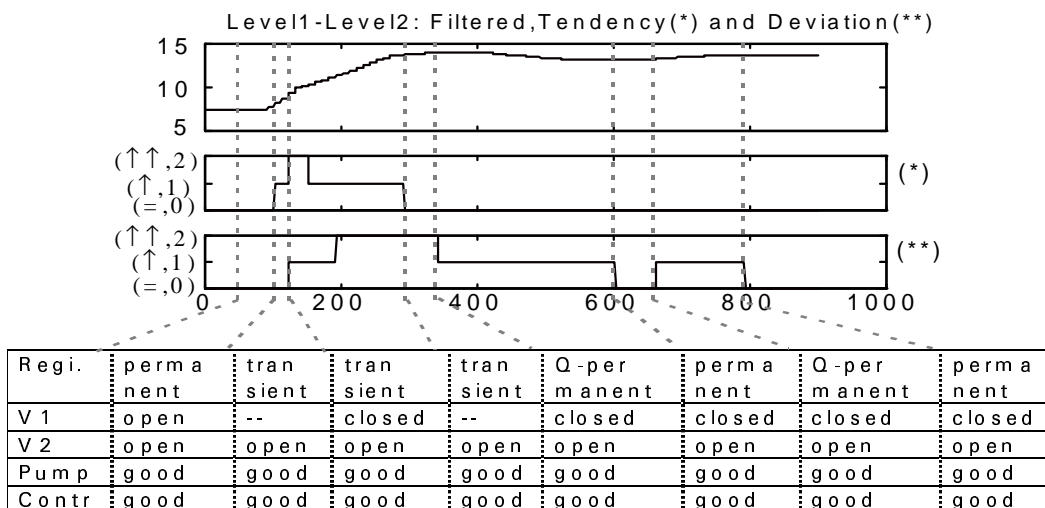


Fig. 6.10 Diagnostic evolution (printed messages), when valve 1 is closed.

Rules are arranged according to the CEES syntax as shown in Fig. 6.11. They can use both numerical and qualitative information as facts, and the certainty associated to these facts is combined when the rule is fired. The deduced *facts* inherit the resulting certainty.

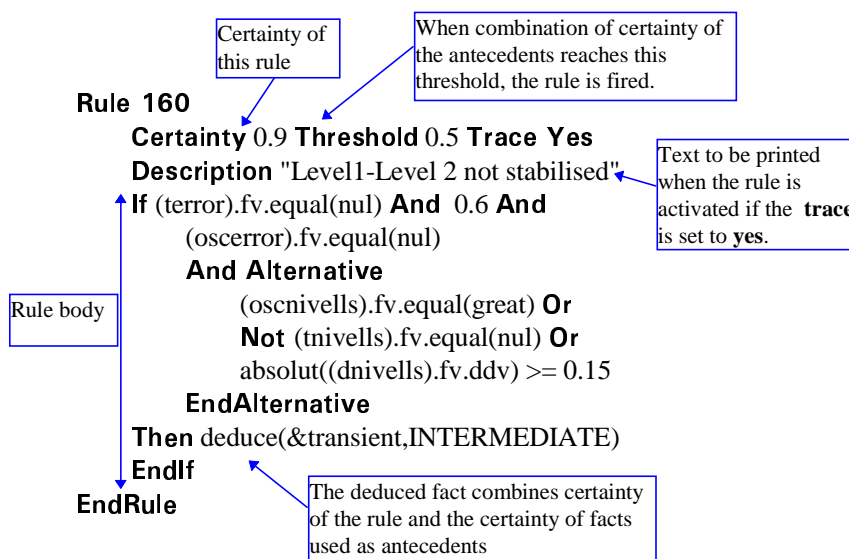


Fig. 6.11 CEES rule.

This example has been used to demonstrate the benefit of using *abstraction tools* for interfacing process variables and ES. The *abstraction tools* are used to obtain significant information at higher abstraction levels (qualitative information) from numerical signals. The kind of tool to use in each development will be different and the goal is not to build a dedicated *abstractor* for each application, but to choose the best for each application from an *abstraction* ToolBox. This simple application shows that different information can be used from a single signal. Thus, it will be better if all the information abstracted from this signal will be encapsulated in *object-variables* and accessible without using a dedicated output line for every attribute. In this case the realisation will be as it is depicted in Fig. 6.12.

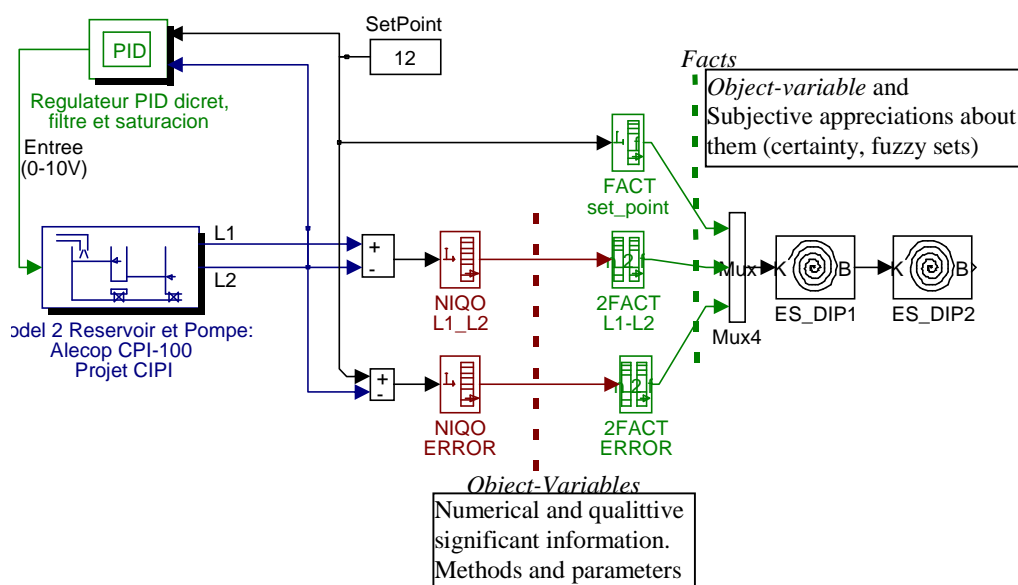


Fig. 6.12 Object-Variables embed abstraction tools to provide ES with necessary information (numeric and qualitative) from signals.

The benefit of using this architecture is to reduce time in the integration task and to design new *abstraction tools*. Although a large set of *abstraction tools* have been developed and tested, they are not yet implemented under *object-variables* and only simple ones are translated to this architecture.

6.3. Qualitative estimation of a variable.

In the following paragraphs an example of how the representation language ALCMEN is used to estimate qualitatively a variable. After the numeric to qualitative conversion using the filtering approach proposed in ALCMEN, the qualitative process variables are combined using static and dynamic operators to

obtain a new qualitative variable. This example has been presented in [Melendez et al. 1996a] . A simulated process and controller is used in this example, namely the previous described laboratory plant composed by two coupled tanks with a PID controller that regulates the level of Tank_2 (See Fig. 6.1 in previous subsection).

In this case, the goal is to deduce roughly the qualitative evolution of the level of tank_1 (Level 1) by means of regulator input and output variables (control and level in tank_2). It is presupposed that not information is available about pipes calibre and diameter or shape of tanks. Information about the process is reduced to ranges of process variables. This is 0-10 volts for control variable and 0-31 cm for tank levels. Level 1 from the simulated plant will be used, exclusively, to validate results obtained from qualitative relations used to deduce the qualitative Level 1 (L1), as it is explained in the following paragraphs.

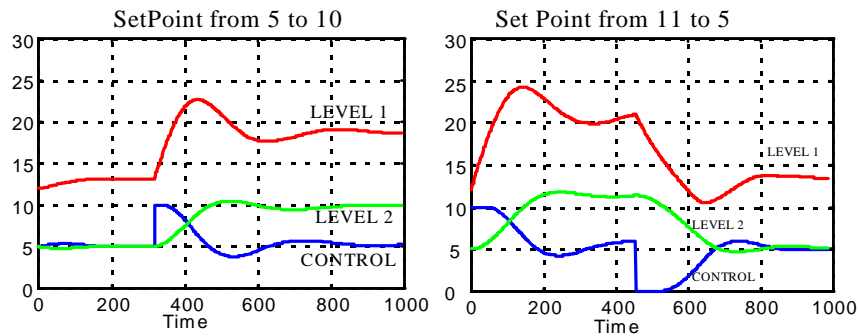


Fig. 6.13 Evolution of signals when changing Set-point.

For this purpose, control and Level 2 are used as numerical process variables. They are converted into a qualitative representation by using a block for building *object-variables* with the ALCMEN filtering method (C and L2 are the qualitative representation of control and level2, respectively). ALCMEN blocks will be used to perform simple relationships between the qualitative representation of process variables in order to deduce the Level 1 evolution. According to physics laws and simple dependencies from common sense, simple qualitative dependencies can be used to estimate level 1 dynamics (L1). Variations of the level of tanks (ΔL) are produced because a change in the input (Q_{in}) or output flow (Q_{out}) of the liquid. Then, level 1 is interrelated to its input (Q_1) and output (Q_{12}) flow, at any sample time (k), by (α operator means a kind of direct dependency between both sides of the equation) :

$$\Delta L1(k) \alpha Q1(k) - Q12(k) \tag{Eq. 6-1}$$

$$L1(k) = L1(k - 1) + \alpha \Delta L1(k) \tag{Eq. 6-2}$$

Thus, the problem of representing level 1 is reduced to deduce both the input (Q1) and output (Q12) flow of tank_1. Q1 is directly related to the control signal considering the motor-pump with rapid response :

$$Q1(k) \propto C(k) \quad \text{Eq. 6-3}$$

And Q12 depends only on the difference of levels. If both levels are equal, then there is no flow and the variations on one level provoke a proportional variation in the flow. This is expressed in the relation :

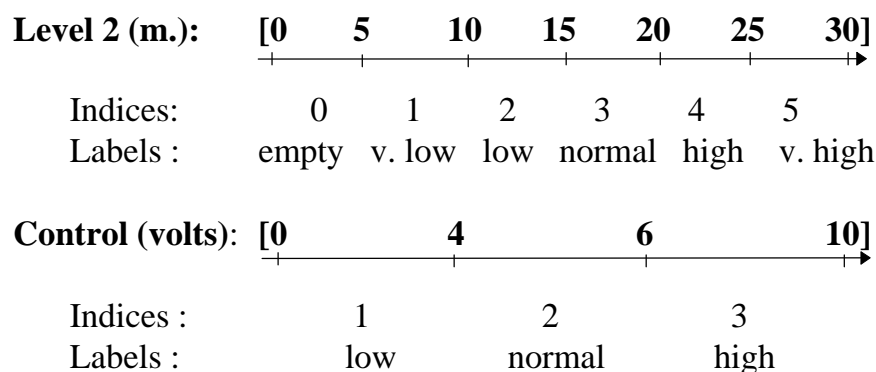
$$Q12(k) \propto L1(k) - L2(k) \quad \text{Eq. 6-4}$$

Using ALCMEN representation, these simple 'equations' can be reproduced. Level 2 and control are defined as numerical variables, and filtering is applied for obtaining a qualitative representation (L2 and C) in their lexical domain (Ls, Lc). The lexical domains Ls and Lc are defined :

$$Ls = \{(\langle \text{empty} \rangle, 0), (\langle \text{very low} \rangle, 1), (\langle \text{low} \rangle, 2), (\langle \text{normal} \rangle, 3), (\langle \text{high} \rangle, 4), (\langle \text{very high} \rangle, 5)\}$$

$$Lc = \{(\langle \text{low} \rangle, 1), (\langle \text{normal} \rangle, 2), (\langle \text{high} \rangle, 3)\}$$

Numeric limits for the crisp zones of their ranges are selected to design each *label*, according to the observation referred to normal operation mode. For example, permanent regime is established with control variable around 5 volts. Then, *normal* label is selected to design this zone (4-6 volts) and inferior and superior regions are labelled as *low* and *high*.



And *filtering* is applied

$$L2 = F(\text{level2} / 0,5) \quad \text{Eq. 6-5}$$

$$C2 = F(\text{control} / 1,3) \quad \text{Eq. 6-6}$$

Using these qualitative variables, C and L2, the evolution of L1 is deduced at any sample time, **k**, according to the previous relationship by following the qualitative ALCMEN dependencies :

$$Q1(k) = C(k) \tag{Eq. 6-7}$$

$$Q12(k) = L1(k) - L2(k) \tag{Eq. 6-8}$$

$$\Delta L1(k) = Q1(k) - Q12(k) \tag{Eq. 6-9}$$

$$L1(k) = L1(k - 1) + g \cdot \Delta L1(k) \tag{Eq. 6-10}$$

Using ALCMEN block representation in the CASSD framework, previous relations are implemented as is depicted in Fig. 6.14, where blocks labelled as L2 and C, encapsulate numeric and qualitative data in their respective *object-variables*. *Filtering* (embedded in the same *object-variable*) has been used as numeric to qualitative interface. The block labelled as O6sr_di is used to estimate the qualitative difference (comparison) between L1 and L2, and the output is supplied to O6co_ge. This block performs the difference between the input and output flow, but due to the fact that different lexical domains are used in this relationship, the lexical difference can not be used. In spite of this, drawback, a difference between both magnitudes can be calculated using tables. In this case, block O6co_ge has been used.

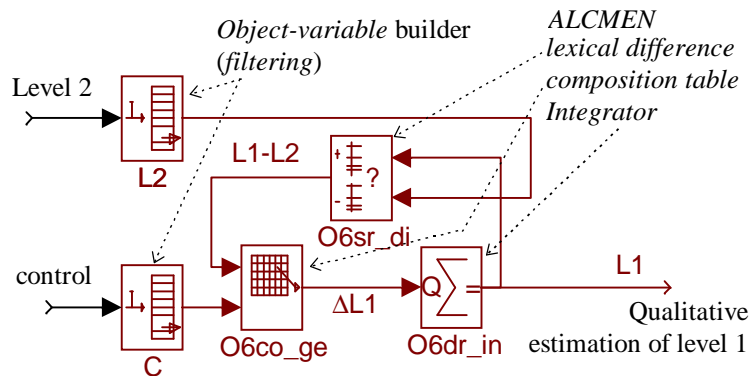


Fig. 6.14 using ALCMEN for qualitative deduction of level 1 (L1)

Block **O6co_ge** is defined to calculate a modified difference at same time that some non-linearity is added to this operation. The table used is represented in Fig. 6.15.

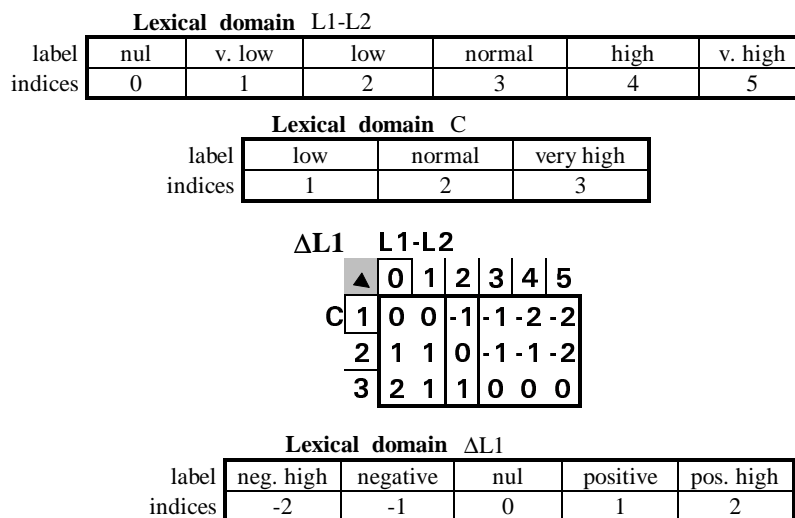


Fig. 6.15 The composition block is used to calculate a modified difference between two qualitative magnitudes.

Level 2 can take any value between maxima and minima, depending on the set point value. The evolution of Level 1 is always tied to Level 2. Therefore, the zones used to qualify Level 2 and Level 1 will be analogous; moreover, the lexical domain must be the same for both variables. This supposition has been taken into account when using lexical difference between both variables.

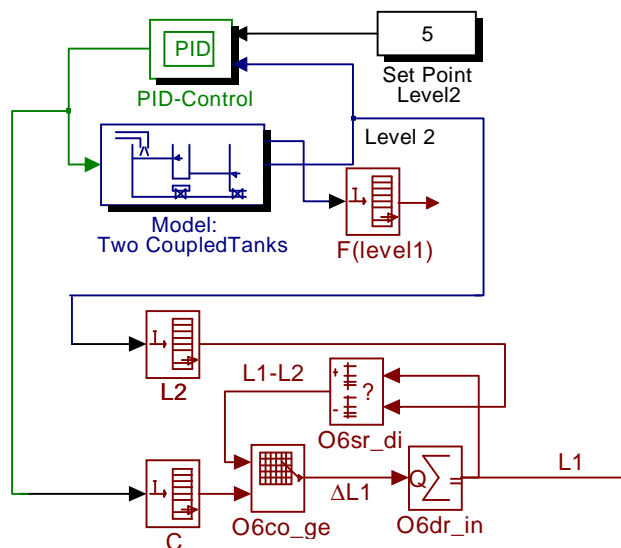


Fig. 6.16 Qualitative observer for estimating L1 using controller input and output (control and level 2).

To test the benefits of this qualitative observer, the result obtained has been compared with the *filtered* value of the simulated level1. This task is done by the block labelled F(label 1) in Fig. 6.16. Both behaviours are depicted in Fig. 6.17 when changes in the set point are ordered. Note that L1 and F(level1) are

coincident in a permanent regime. Dynamics are also preserved although some undesirable transition between indices is given. This is due to the nature of the qualitative simulation with crisp transitions between zones. For example at the beginning of Fig. 6.18, level 1 and level 2 have undesirable transitions because limit of zones election is exactly the value in the steady state (5cm. for level 2).

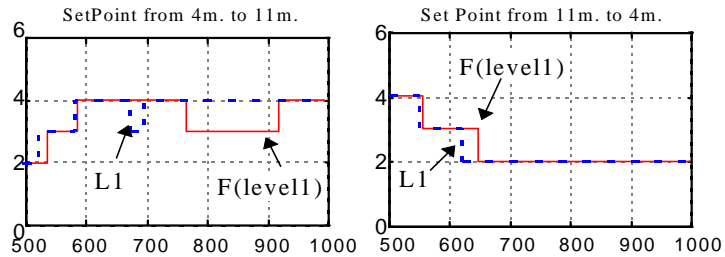


Fig. 6.17 Comparison of qualitative level 1 (indices), L1, obtained with ALCMEN (**dotted line**) and, F(level1), from N/S conversion from direct measure (**solid line**).

In a larger and more accurate simulation, several situations can be introduced in the process. For example, process dynamics can be changed by narrowing the fluid pass between tanks. This has been done closing valve V1. The evolution of process variables (level2 and control) and the qualitative observed level1 (L1) are represented in the next figures, when the valve is open (Fig. 6.18) and closed (Fig. 6.19) and the set point is changed every 750seg. Process dynamics are different because of this introduced structural change (V1 closed).

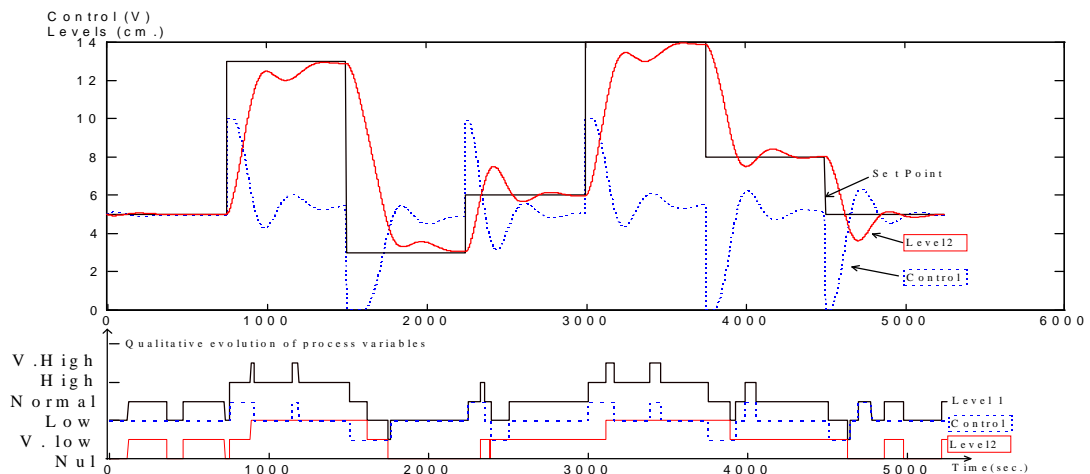


Fig. 6.18 Evolution of L1 when set point is changed every 750s. Valve V1 is open.

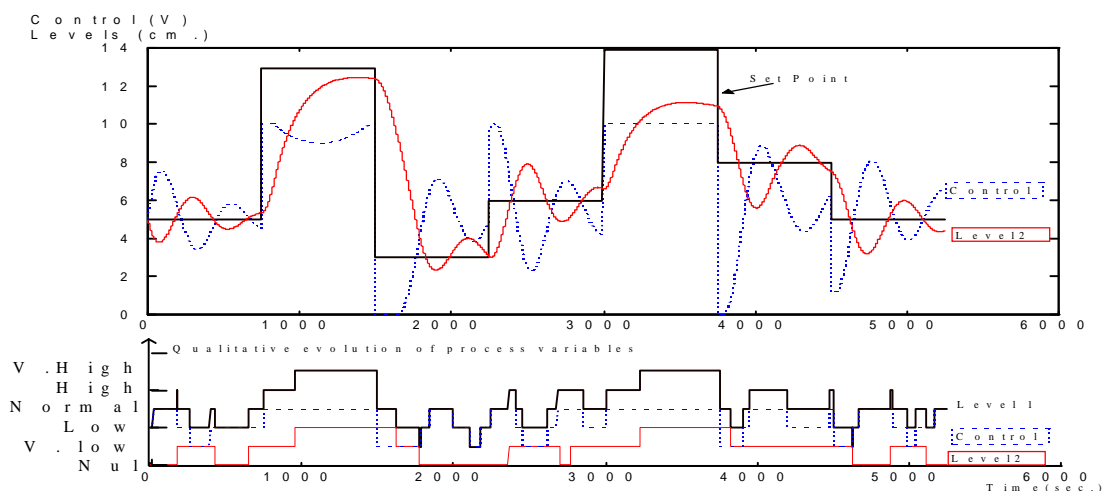


Fig. 6.19 Evolution of L1 when set point is changed every 750s. Valve V1 is closed

For instance, this qualitative machines could be used to provide ESs with meaningful information, avoiding to perform relations between variables into the ES. In the example, a non-accessible process variable is roughly deduced. Although the evolution of L1 is not very precise, it seems sufficient to deduce some changes in the dynamics of process.

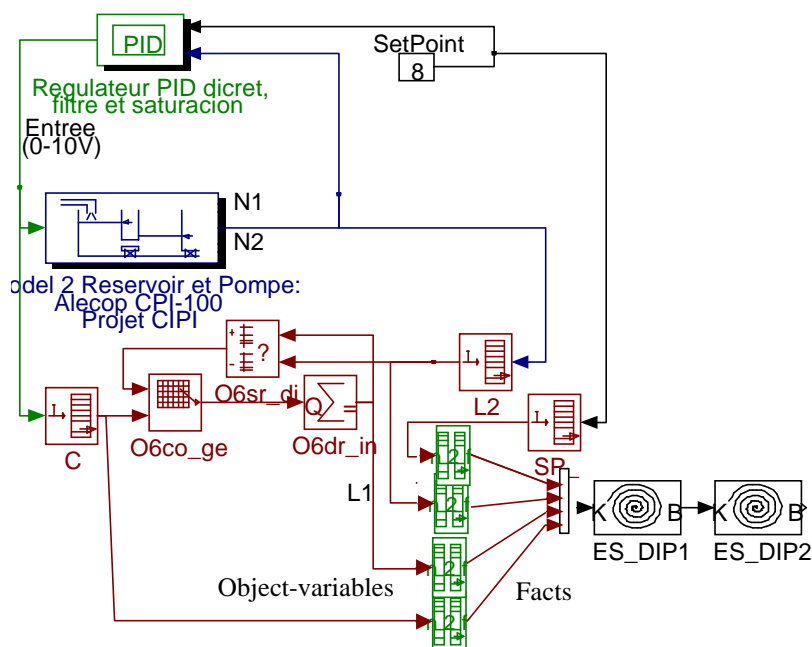


Fig. 6.20 Fault diagnosis system dealing with *object-variables* and qualitative estimated variables.

Next step is to deal with this qualitative deduced data, using it for reasoning tasks. For instance the previous example of fault diagnosis system of two coupled tanks can be reproduced, using control and level 2 as accessible process variables and using the ALCMEN relationship to roughly estimate the level 1 dynamics. The configuration for this example is depicted in Fig. 6.20 ALCMEN blocks are prepared to deal with *object variables*, using only their stored qualitative information and rebuilding new *object-variables* at their output. This *object-variables* are stored into *facts* to be supplied to the ES.

This configuration is being tested. The main difficulty is using the qualitative deduced information because of the crisp zones. Changes in the qualitative estimated variables produce sudden transitions from one diagnostic to the opposite one. Fuzzy capabilities of ES are being used to smoothen these transitions, but this is not always satisfactory.

6.4. Qualitative estimation of temperature in the furnace of a Municipal Solid Waste Incineration plant.

Simple qualitative relationships have been used to predict the evolution of the temperature in a furnace for the incineration of a municipal solid waste (MSW). This is a real plant established in Girona (Spain). The activity of the plant is to burn MSW coming from the metropolitan area for producing electricity. MSW are used as raw material to heat water for producing the steam flow that moves a turbine. MSW are burned in a furnace to produce high temperature flue gases (1000 °C) that are used to heat the water. The main difficulties in controlling the furnace, and consequently, the evolution of the output temperature, are due to the diversity of calorific value and humidity of the input (MSW). This temperature must be controlled to assure a regular steam flow.

This section is reproducing the partial work done to determine the influence of the quality of MSW on the evolution of the temperature of the furnace's output fumes. The goal is to establish a qualitative model to predict the output temperature of the furnace. This work is not finished yet, but partial results are reproduced here to demonstrate the capabilities of using block representation of ALCMEN relationship in qualitative modelling.

6.4.1. Plant description : the furnace.

The main component of a furnace used to burn MSW is the grate. In the plant of Girona, this is a Martin grate with two inclined conveyors with backward movement. An hydraulic ram is used to activate the grate to produce a to-and-

from movement to make the refuse to advance while it is burning. Ashes are separated by gravity. Fig. 6.21 shows the situation of the main components in the furnace. The MSW passes through three different areas in the grate. The first one is the drier, where the refuse is heat at high temperature to decrease its humidity. Later on, in the combustion area, the refuse is burned and transformed in ashes and slag. Finally, the fire extinguishes in the third area and the slag is ejected. Although the refuse is dried in the first area, when it goes trough the second area, the calorific value and the humidity of this material is not constant, and it is difficult to control the furnace to assure an uniform flue gas.

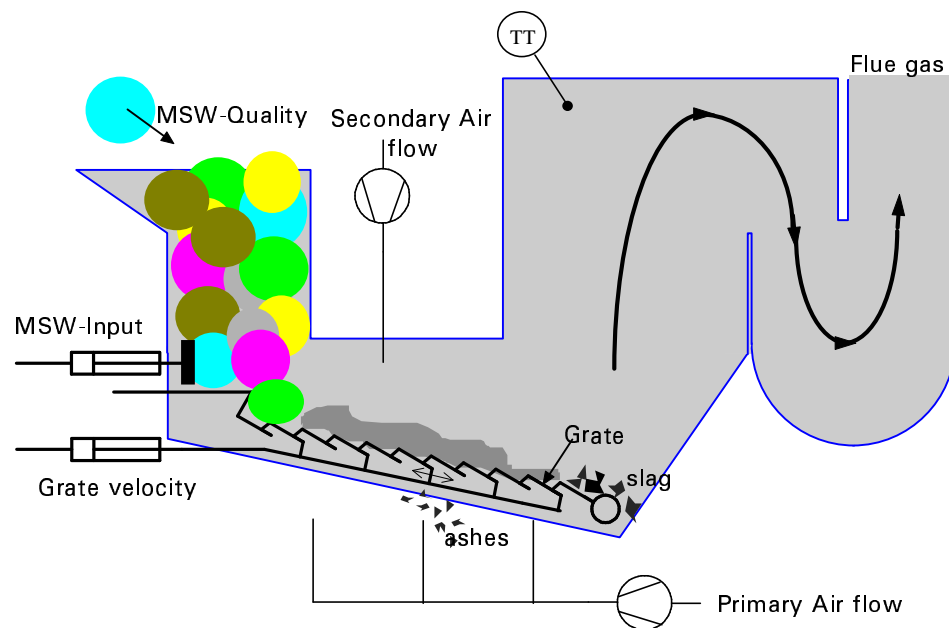


Fig. 6.21 Furnace for incinerating MSW.

The burning capacity of this kind of ovens is basically determined by the following variables:

- Quality of MSW (calorific value, humidity, density, composition, volatile components, ashes).
- Temperature (in the oven, of the output gas, combustion air)
- Air flow (In the primary and secondary circuit)
- O₂ contents.
- Steam flow.

These variables are strongly interrelated. It is difficult to establish a dynamic model of the furnace, because some of these variables are difficult to measure or present a non-uniform distribution in the oven. Despite that the exact dependencies are not quite known, the influence of some variables on the evolution of the furnace is clear and, in fact, control is basically performed actuating on three variables:

- Combustion air flow.
- Grate movement. Velocity and displacement.
- Quantity of input waste.

Actions on these variables are decided according to the temperature evolution, MSW-quality and steam flow in the normal operation mode. O₂ contents are taken into account in start and stop modes. The first variable, combustion air flow, is automatically regulated according to the set point established. And the second and third variables are manually tuned according to the temperature variations (in the furnace and steam flow) and MSW-Quality.

6.4.2. About MSW quality.

Municipal waste include a wide range of materials, such as rubbish, vegetables, fruits, paper, plastic, glass, tin and so on. All of these materials arrive to the plant mixed in different proportion according to the point of origin in the city (market, city centre, residential area, service enterprises, ...) and the time, and they are stored mixed within a bunker to be supplied to the furnace. Consequently, the evolution of the furnace is strongly dependent on the quality of these MSW, which is a parameter difficult to measure or estimate. The influence of humidity and calorific value of the material in determining the quality of MSW for incineration as depicted in Fig. 6.22 is obvious, but it is also difficult to measure both parameters since it is not an homogeneous mixture.

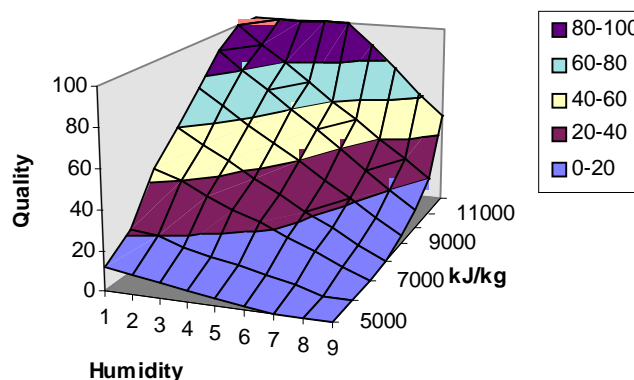


Fig. 6.22 Quality of MSW.

Therefore, the charge of the furnace is always assisted by an human operator who observes the composition of the row material and decides about its quality. This information is used to adjust the grate movement and the input quantity.

6.4.3. Qualitative estimation of temperature.

The main problem in controlling the furnace described before, is the kind of row material used as input and the inherent difficulty to measure it. As a consequence, the temperature measured of the flue gas, used in the steam production, suffers sudden variations as shown in the register of Fig. 6.23. The output temperature is not very useful in controlling the furnace because of the thermal inertia of the furnace. Therefore, it seems better to undertake some action to anticipate process dynamics.

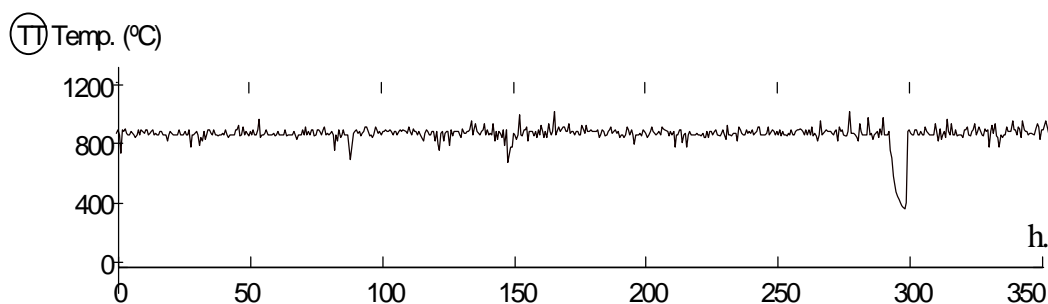


Fig. 6.23 Register of flue gas temperature in the furnace.

Thus, the challenge is to take benefit of the operator experience and process engineer knowledge and to try to predict the evolution of the furnace. Expert engineer established the influences depicted in Fig. 6.24 between process variables and MSW, when a positive variation is given one of the MSW parameters depicted in the figure :

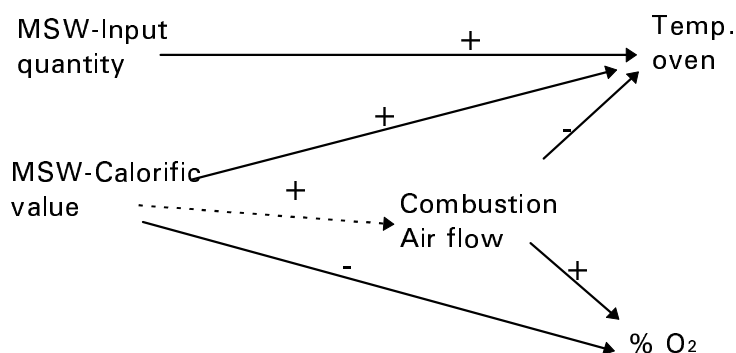
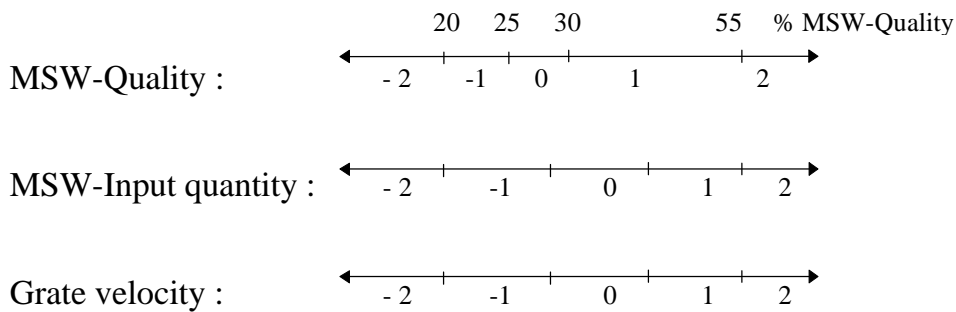


Fig. 6.24 Influences between process variables inside the oven. Dashed arrow shows the action to perform and solid arrows are given as process restrictions.

With the goal of using expert observances, process operators were asked to fill a table to roughly classify the quality of MSW input, according to their expert criteria, and the changes performed in the furnace parameters (grate velocity and variation of MSW quantity). This information has been used to estimate the temperature variations in the furnace without taking into account the regulation of the combustion air flow. ALCMEN has been used with this goal to establish a graph and to simulate these dependencies. It has been tested with the data of a fortnight during the summer. The *lexical domain* for the all the variables involved is defined with two positive labels for positive increasing (indices 1 and 2) and two negative, for decreasing situations (indices -1 and -2) from the normal mode (index = 0). The MSW-Input quantity and grate velocity are obtained directly as qualitative variables according to the position of commands selected by the operator. On the other hand, the indices corresponding to MSW quality are estimated to be centred between 25% and 30% because the period of year during which the measures have been acquired, i.e. summer, is characterised by wet garbage (due to the big amount of fruits and vegetables). Then, the indices used are :



And the ALCMEN relationship in the CASSD framework, modelling the positive influence of these variables in the evolution of temperature, are depicted in Fig. 6.25.

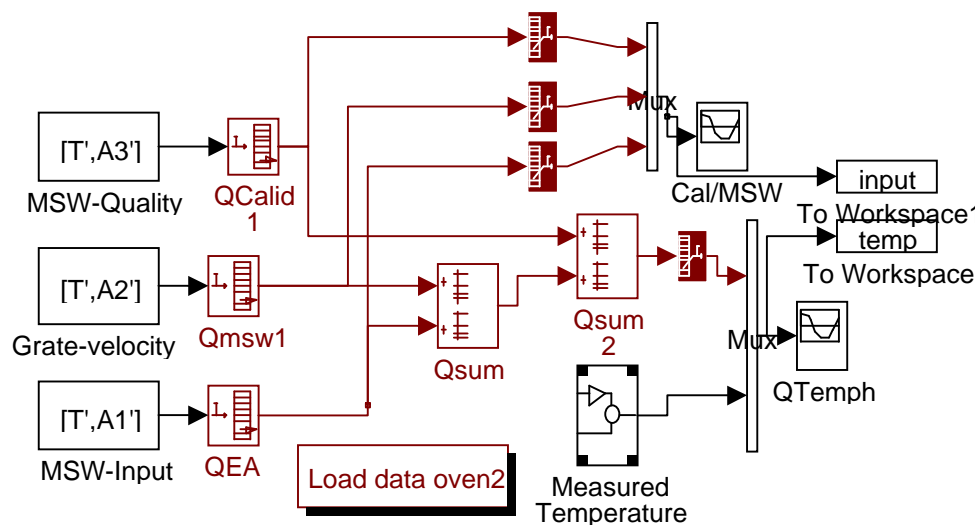


Fig. 6.25 Estimation of furnace temperature using ALCMEN blocks in the CASSD framework.

Two simple blocks (Qsum and Qsum2) have been used to add the MSW influence (Quality, Grate-velocity and quantity of MSW Input). The behaviour of this qualitative estimator is tested using qualitative data supplied by the operators as input and the register of temperature in the oven. The temperature evolution has been smoothed and qualified in 5 zones around the average value. The comparison of both can be observed in Fig. 6.26.

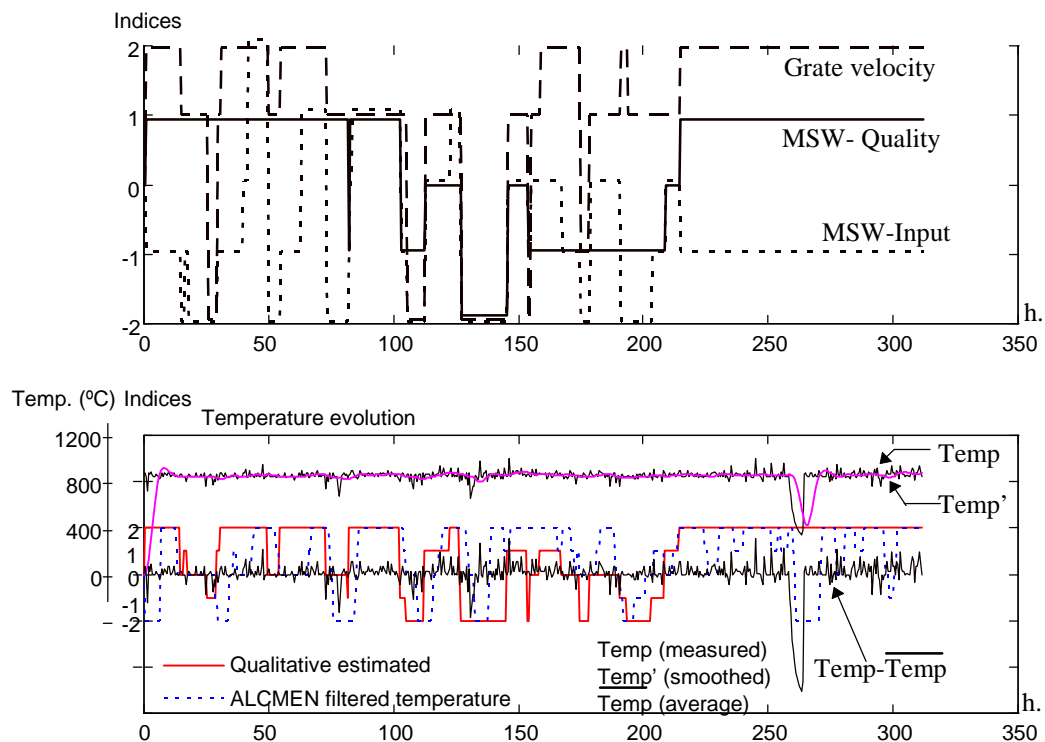


Fig. 6.26 Qualitative estimation of temperature in the furnace.

The validity of these results is up to 215 hours because after that all input remains constant (it was because an emergency occurred and operator left the last value). When the time equal 260h. the temperature is decreasing very fast due to a stop in the plant and which is not reflected in the input set of values. Then, the final set of data must be rejected. The evolution of qualitative estimated temperature and the qualification of real data have similar behaviour. There does not exist a perfect matching between both, but large oscillations are more or less detected. A delay between both can be observed, because the temperature transmitter is placed very far from the input of MSW, and the data given, refers to the input MSW. Moreover, MSW quality is given outside of the input hopper. Although the general evolution is satisfactory, it must be contrasted with more

data and added with the influence of the air flow although it remains constant when selected a working point.

6.5. Validation of a knowledge base when designing an ES for fault diagnosis.

In the process of developing an ES for fault diagnosis the validation of KBs is very important. When the design admits a modular partition of the ES in specialised tasks, the validation becomes easier. This example is to show how a CASSD framework can be used to validate one module of the ES. In this case, the application was built as a stand-alone application linking the ES LabCEES with a SCADA system monitoring a central heating plant used for training. The CASSD framework was used to test and tune rule parameters (limits, certainties, ranges) in the ES using real data stored in the SCADA system.

A synoptic of this training plant, extracted from the SCADA system, is reproduced in Fig. 6.27. Three boilers (one petrol boiler and two gas boilers) heat water in the primary circuit and this water is pumped by two motor-pumps. A heat exchanger is used to transmit the heat to the secondary circuit. The primary circuit is auto-regulated by means of three ways valves to avoid returning water to be excessively cold. This situation could cause damage in the boiler.

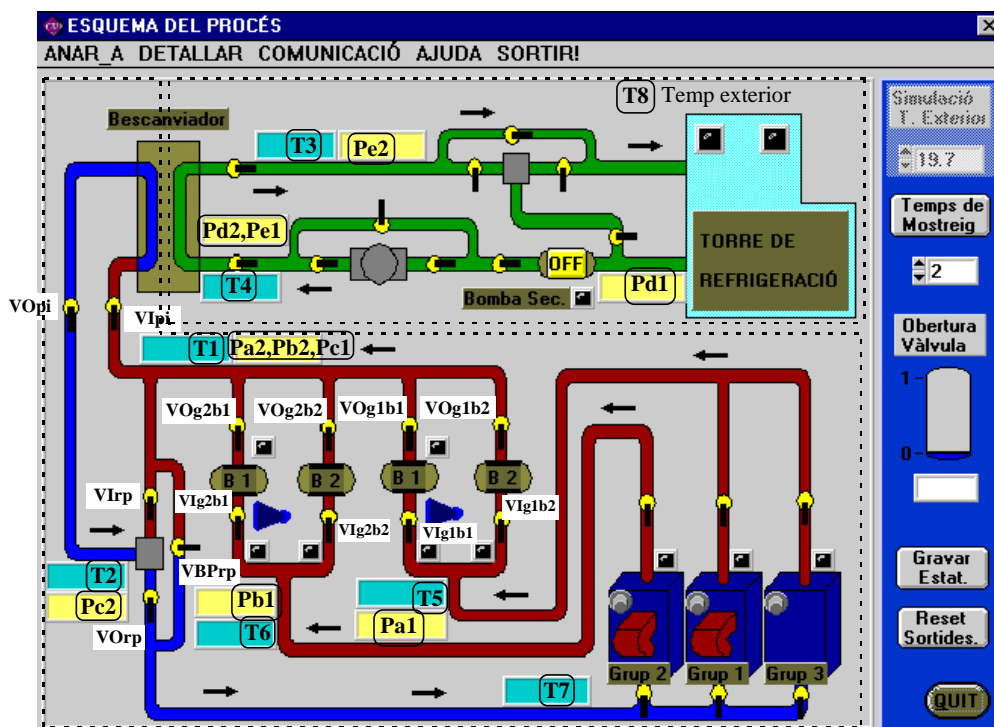


Fig. 6.27 Window from the SCADA system.

The role of the ES is to supervise the primary circuit to prevent obstructions in the pipes by using measured pressures and temperatures. The obstructions are simulated by closing valves at different points.

In this case, some faults (obstructions) have been reproduced by closing valves and storing data using the SCADA system. These data have been analysed in the CASSD framework. Simple ES modules have been developed to prevent fault situations. The idea is to develop simple supervisors specialised in detecting a single fault. Measures are directly converted into *facts*, and supplied to the ES module to reason about them. One of these modules is tested in Fig. 6.28 to diagnose about status of valve labelled as VBPrp. Its normal state is closed and misbehaviour is simulated closing this valve. In this case fault detection is difficult because the presence of a regulated valve in parallel with this one that compensates the perturbation introduced when VBPrp is open. Additional drawback in this process is due to instrumentation restrictions. Measures of pressures are only available in pairs. Thus, we can only get simultaneously the pairs of pressures labelled as : Pa1-Pa2, Pb1-Pb2, Pc1-Pc2, Pd1-Pd2 and Pe1-Pe2.

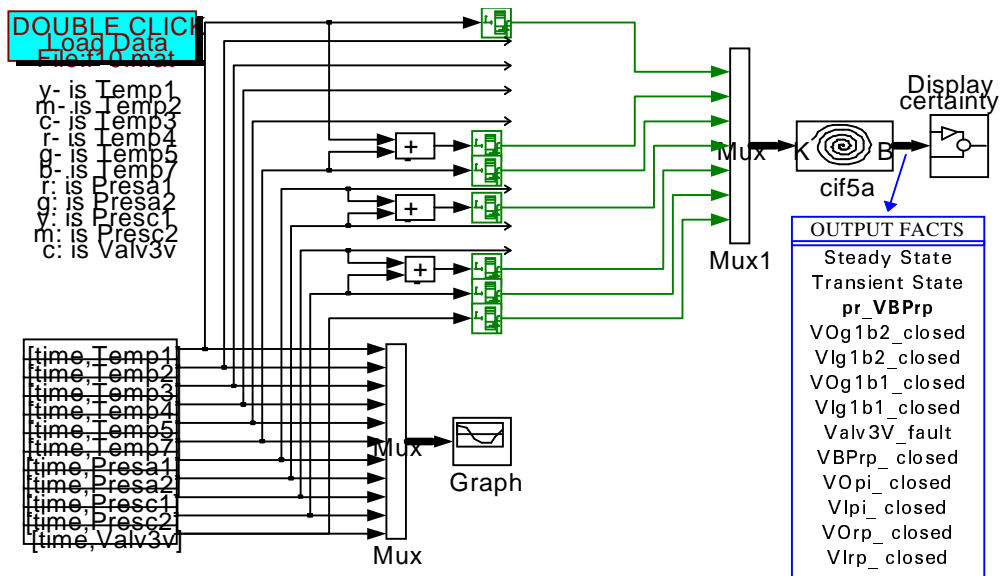


Fig. 6.28 Testing CEES module with real data.

In this kind of process, primary circuit is closed. It means that an obstruction or a similar functional misbehaviour is reflected in the pressures sensors in similar way independently of the localisation of the obstruction. Moreover, the evolution of Pa1 and Pa2 is similar for all obstruction and is better to use Pc1 and Pc2. Thus, following paragraphs are related to this second fault in Fig. 6.29. Therefore, if it is difficult to detect a fault using only pressures (Fig. 6.29), the

use of temperatures (T1-T7) and additional information about openness degree of the three ways valve (Fig. 6.30) is necessary to locate the fault (diagnose).

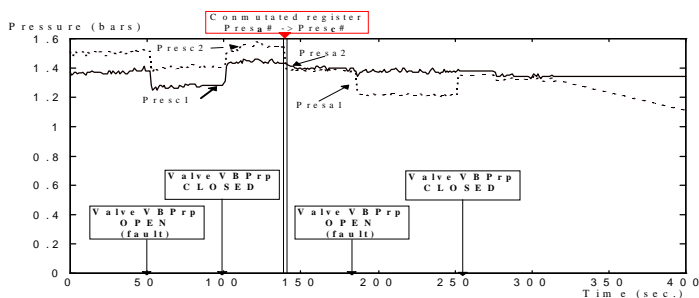


Fig. 6.29 Measured pressures Pa1 and Pad and Pc1 and Pc2.

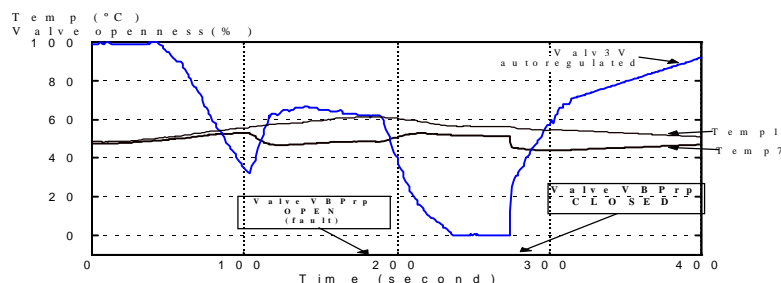


Fig. 6.30 Evolution of the 3 ways valve and temperatures T1 and T7.

In this case the difference of temperatures T1 and T7 related to the position of the valve (openness degree) is used to differentiate valve VBPrp from valves VOg1b2 and VIg1b2 during the uncertainty time. The evolution of output *facts* certainty in the successive approaches of rules can be observed in the graphs of Fig. 6.31. When additional relations are taken into account, faults can be isolated.

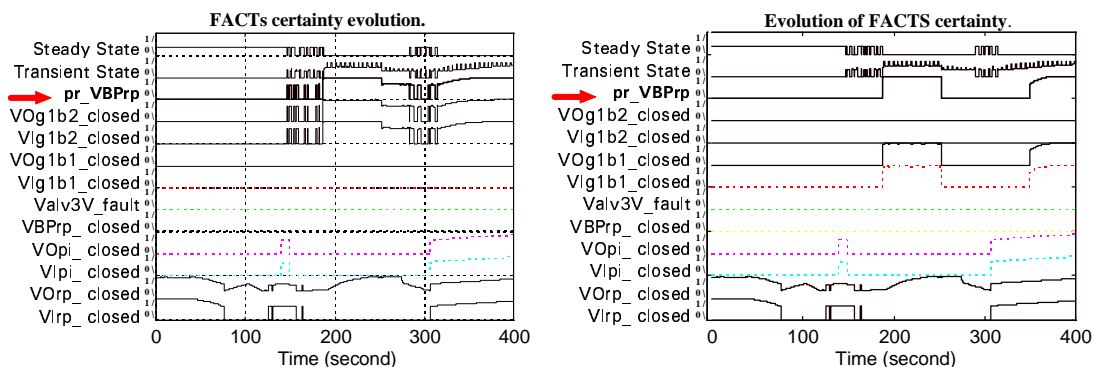


Fig. 6.31 Evolution of facts certainty. On the left, when only pressures is taken into account, and on the right if additional reasoning about T1-T7 and degree of openness of Valve3V is performed.

After validation of each module rules they can be interconnected to validate the whole ES. In this case, the CASSD framework was only used in validation of partial KB modules using acquired data. The final implementation was performed in LabCEES connected with the SCADA system by means of dynamic data exchange (DDE) because difficulties in acquiring pairs of pressures. This task is managed by the SCADA systems according to data asked by the ES. This final implementation is described in [Martinez, 1997]. This is the report of a complementary work of this thesis.

7.

Conclusions and Future Work

Global conclusions are summarised here, while partial conclusions have been added at the end of each chapter. Main points developed in the work are emphasised and commented in the first subsection while future work and open lines derived from this work are added in the second subsection.

7.1. Conclusions.

Despite of the great number of tools and techniques to assist engineers in the design of supervisory systems there is not an unified theory and methodology to follow in such developments. While analytical methods have been proved to be successful in the first stage of supervisory systems, i.e. analytical model-based fault detection, following steps are always needed of knowledge-based techniques. Moreover, the difficulty in obtaining accurate models of process, causes that expert knowledge will be necessary in many applications even in fault detection. Consequently, AI techniques must be used together with numerical facilities for representing and processing expert knowledge into computers. At this point, main difficulties are in managing different kind of information, such as data (numerical, qualitative, symbolic or logic) and knowledge, and in integrating the tools needed for this purpose. It would be very convenient of disposing of all of these tools, from the AI domain and numerical methods, available in the same

framework to facilitate design task. Additional tools for representing, evaluating and testing are also necessary.

Major effort in the work has been focused on obtaining a framework for assisting supervisory tasks in the design time. The proposal is centred on integrating such tools that are required for dealing with expert knowledge. Consequently, the presented framework has been developed by adding artificial intelligence techniques into a CACSD framework. Several mechanisms have been developed to support engineers to represent knowledge in a graphical user interface. The benefit of using a graphical user interface simplifies learning and development tasks.

MATLAB/Simulink has been chosen as starting point because its openness, graphical user interface (block based graphical user interface), proximity to control community (control ToolBoxes and extended use) and numerical facilities (representing capabilities and signal processing ToolBoxes). The same proposal could be tested in another CACSD framework with similar capabilities. In fact, G2 was tested at the beginning, because its more extended use in industrial environments, but it was left behind because its poor support for signal analysis and processing, needed in the interface between expert system and process (*abstraction tools*) and difficulties for dealing with uncertainty and imprecision in the rule-based ES.

MATLAB/Simulink openness has given the opportunity of testing OOP as integration technology. Integration has been done at level of process variables. The concept of *object-variable* has been introduced in this thesis as objects dealing with dynamic data (process variables). *Object-variables* are used to encapsulate all necessary information related to variables, such as numerical data (from sensors, simulations or other numerical sources), qualitative data (qualitative description of process variable, landmarks, and so on), methods (to obtain significant numerical or qualitative data) and parameters (supplied by users). Specific Simulink block has been designed to deal with *object-variables* building preserving some of the commonly accepted properties of objects such as inheritance, encapsulation and data hiding. Some attributes of the *object-variables* are dynamically actualised at each sampling time (process related information) while others can be changed asynchronously by users (parameters). *Object-variables* are used to obtain a plain integration of tools. The objective consists in that any tool can access process variables in the best representation (acquired signal, abstracted significant information or qualitative representation).

A set of tools from the AI (*Abstraction tools*, ALCMEN, CEES), have been selected to be integrated in this framework to assist developing of supervisory

structures avoiding interfacing problems. Tools have been adapted to have a Simulink block representation with capabilities of dealing with *object-variables*.

Abstraction tools or *abstractors* are the set of algorithms tested to be useful as numeric to qualitative interfaces. These are the algorithms to embed into *object-variables* to provide the adequate representation of process variables at each tool. Multiple algorithms can be used with this purpose and the goal is to dispose of a library of *object-variable* builders with different capabilities for qualitative representation and abstraction of significant information. The election of one or other for any task depend on both, the process variables and the application to be developed. Until now, only two different types of *object-variables* have been build according to two abstraction methods (*histogrames* and *ALCMEN filtering*) but several types of abstractors are being tested to be added as *object-variables*. Special emphasis is done in methods based on temporal window and triangular representation. The first, because they give smoothed representation of signals behaviour that avoid undesired changes in the qualitative representation of signals and they offer the possibility of event generation. Triangular representation is though to provide qualitative representation of signals in episodes.

ALCMEN has been chosen as a representation language for representing simple qualitative relationship and dependencies. It is not conceived as a simulation language, but its utility as qualitative observer, to roughly deduce or estimate not accessible process variables, has been demonstrated. Intuitive relationship, or more formal qualitative models, can be build to be driven with numerical variables. ALCMEN has been tested with success in several applications because its simplicity and easy use and tuning. A more complete and formal qualitative simulation language, is thought not to be useful in supervisory tasks because the imprecision and uncertainty degree introduced after several iterative steps.

The necessity of production rules for knowledge representation has been solved with the shell CEES. It has been selected because its capabilities of dealing with uncertainty and imprecision in both data and rules. CEES is based on the object oriented approach where ES are objects reasoning about *facts*. *Facts* are build from *object-variables*. ES and *fact* builders have a block representation in the framework for an easy use.

Nowadays, the development of object oriented technologies offers a good solution to structure information. This is very useful when developing large applications. This is the case of supervisory systems where several tools can be applied but all of them dealing at different abstraction levels. When this tools comes from the artificial intelligence domain, the problem of managing and

interfacing them increases and the manipulation and reformatting of data become necessary. In such applications the use of objects allows both data and tools encapsulation facilitating its use. In this work, data is encapsulated at two different levels as *object-variables* or *facts*. Data flow from one tool, represented by a block, to another block. The interface between data levels is performed by 'object-builders' blocks. Other blocks give direct access to any field in the object to work as numerical Simulink data.

Capabilities of using this framework in several steps of the supervisory chain have been tested in several applications. ALCMEN has been validated as useful tool for building qualitative observers. It has been used to estimate a process variable by using qualitative representation of available numerical process variables and performing qualitative relationship between them. In this case ALCMEN blocks takes the qualitative representation of numerical variables from the *object-variable* structure. Simple qualitative models based on variables interaction have also been built. The use of ES interfaced by *abstraction tools* to provide adequate representation of dynamic process variables through time has also been successfully tested. Main difficulties are in choose adequate parameters to tune abstraction tools to interface description of variables given by experts with numerical variables coming from process. The use of qualitative observations provided by ALCMEN relationship, together with the ES is being proved. Difficulty increases in this case because the imprecision (in time and magnitude) of qualitative observed variables.

Experiences with this framework have demonstrated its usefulness for developing supervisory strategies according to rapid prototyping methodology. This is the iterative procedure based on the steps of concept development, knowledge implementation, testing and analysis to reach the desired goals.

To summarise the main results presented in this thesis consist of the following developments :

- Conceptual study, analysis, development and implementation of qualitative and symbolic knowledge processing tools for computer aided supervisory system design .
- Incorporation of object oriented technology into MATLAB/Simulink environment.
- Implementation and incorporation of ALCMEN, set of tools for dealing in representation and reasoning with qualitative information, as a set of blocks as a Simulink ToolBox.
- Implementation and incorporation of rule-base object-oriented methodologies into the MATLAB/Simulink environment. The expert system CEES ToolBox.

- Study of possible practical applications on the base of several examples presented in the thesis.

All the tools were implemented within the same computational platform. This is an important factor deciding about easy access to user friendly environment for developing practical applications. The implemented tools provide extended capabilities, especially with respect to symbolic information processing which are not accessible in the original system. The augmented in this way extended numerical-symbolic design environment should constitute a new paradigm for Computer Aided Supervisory System Design.

7.2. Future Work.

Necessity of dispose of a framework to design and test supervisory architectures was in the aim of the thesis. It is an actual and open research line. This thesis is only a starting point and it is still open in many points. More emphasis must be put in defining objects hierarchy and architecture for both data encapsulation and tools management. In fact actual framework can be enhanced in following topics :

- The use of *object-variables* has been tested to be useful as numerical to qualitative interfaces. Therefore, a complete set of these *object-variables* must be build as a complete library according to presented abstraction algorithms.
- More complex *object-variables*, as actual *facts*, can build in a hierarchical architecture to represent knowledge about process variables. This will be useful for new tools dealing with knowledge representation and processing.
- ALCMEN, can be improved adding imprecision management in the relationship. Fuzzy logic can be applied in the tables implementation performed. In the same way numeric to qualitative conversion, *filtering*, can be implemented as a *fuzzyfication* and actual qualitative variables can be converted to fuzzy sets.
- The ES, CEES, it is being redesigned to obtain a more flexible architecture to deal with any *object-variable*. The inference mechanism can be improved with backward tracking and multiple step reasoning.
- Additional tools, analytical and AI, must be integrated to deal with *object-variables*. For instance statistical methods or classification algorithms can be used to improve supervisory strategies.
- Data base management could be added to this framework to deal with case-based reasoning. The combination of several reasoning methods with different types of knowledge representation can be useful

for the reuse of stored cases in order to detect specific situations and prevent abnormal operation conditions.

All of these improvements must be done under the new version of MATLAB (v.5) and Simulink (v.1.2) taken into account additional features and capabilities offered by MathWorks. New structures, near to objects, and ToolBoxes such as Stateflow and enhanced user interface controls are in the same line and will improve final result.

Moreover, actual tendencies in distributed application could also be used to implement a framework for designing supervisory applications. With this aim integration capabilities can be extended to be used as an in open framework of a distributed architecture. For this purpose, it is necessary to rebuild the environment according to a reference model. One possibility is to adopt the Object Model proposed by the OMG (Object Management Group) . The benefit of using this model is in the interfacing stage of new and distributed (in the network) tools [Vinoski, 1997]. The CORBA (Common Object request Broker Architecture) specifications adopted by the OMG [OMG, 1995] , details the characteristics and interfaces to access to other object tools. Using this reference model for the supervisory tool can facilitate the design of supervisory structures. *Object-variables* implemented under this model, or higher object structure grouping several *object-variables* can be used as server application to provide all necessary information related to process measures.

8.

Glossary

Acronyms and Abbreviations :

AI :	Artificial Intelligence
ALCMEN :	Automaticians Language for Causal Modelisation for Expert kNowledge
CAD :	Computer Aided Design
CAE :	Computer-Aided Engineering
CASE:	Computer-Aided Software Engineering.
CACE:	Computer-Aided Control Engineering.
CACSD:	Computer-Aided Control Systems Design.
CASSD :	Computer -Aided Supervisory Systems Design
CEES :	C++ Embedded Expert System.
CORBA:	Common Object Request Broker Architecture
ES:	Expert System
ECMA:	European Computer Manufacturers' Association
FIR :	Fuzzy Inductive Reasoning
KB:	Knowledge Base
NIST:	National Institute for Standards and Technology (USA)
OOCACSD:	Object Oriented Computer Aided Control Systems Design.
OOES :	Object Oriented Expert System.
OOP :	Object Oriented programming
QR :	Qualitative Reasoning
SCADA :	Supervisory Control and Data Acquisition.

Packages

G2	Trademark of Gensym
MATLAB	Trademark of the Math Works, Inc.
Matrix _x	Trademark of Integrated Systems, Inc.
Simulink	Trademark of Math Works, Inc.
Windows	Trademark of Microsoft.

9.

Bibliography

- [Aguilar-Martin, 1991a], “ALCMEN, A language for qualitative/quantitative knowledge representation in expert supervisory process control”. *LAAS Report*. Jan, 1991.
- [Aguilar-Martin, 1991b], “Representación simbólico numérica para sistemas expertos de control en tiempo real”, *Curso de Verano Universidad Internacional Menendez Pelayo*, Santander, 1991.
- [Aguilar-Martin, 1993], “Knowledge-based Real Time Supervision of Dynamic Processes : Basic Principles and Methods”, *Applied Control : Current Trends and Applied Methodologies*, Ed. Marcel Dekker, Inc. , 1993.
- [Aguilar-Martin, 1994], “Qualitative control, diagnostic and supervision of complex processes”, *Mathematics and Computers in Simulation*, 36, pp 115-127, 1994.
- [Årzen, 1995], “Experiences of using G2 for real-time Process Control”, in “*Supervision de processus à l'aide du système expert G2*”, Ed. Hermes, Paris.
- [Årzen, 1995b], “AI in the feedback loop : A survey of alternative approaches”, in IFAC Workshop on “*Artificial Intelligence in Real Time Control*”, pp : 207-218, Bled, 1995
- [Åström, Anton and Årzen, 1986], “Expert Control” in *Automatica*, 22 :3, pp : 277-286, 1986.

- [Åström et al., 1993], Åstrom, T. Hägglund, C.C. Hang, W.K. Ho, "Automatic Tuning and adaption for PID controllers- A survey", *Control Eng. Practice, Vol 1, N4*, pp 699-714, 1993.
- [Ayrolles et al., 1995], Ayrolles L., Aguilar-Martin J., Guerin F., "Interprétation Symbolique pour la Supervision Multi-Résolution de Processus Dynamiques", *Supervision de Processus à l'aide du Système Expert G2*, pp 73-90, Ed Hermes, 1995.
- [Ayrolles, 1996], «Abstraction temporelle et interprétation quantitative/qualitative de processus à dynamiques multiples. Application aux processus biologiques», *These de Doctorat Université Paul Sabatier*, Toulouse, France, 1996.
- [Bakshi and Stephanopoulos, 1994], B.R Bakshi and G. Stephanopulos, "Representation of process trends, part III and part IV", in *Computer Chem. Engng. Vol 18, No 4* pp267-332, 1994.
- [Bakshi B.R. et al., 1994], "Analysis of operating data for evaluation, diagnosis and control of batch operations", *Journal of Process Control*, vol 4, n° 4, pp 179-194, 1994.
- [Barker, 1995], Barker H.A., "Open-environmets and Object-oriented methods for computer-aided control systems design", *Control Eng. Practice*, Vol. 3, pp 347-356, 1995.
- [Blankenship et al., 1995], Blankenship G.L., Ghanadan R., Kwatny H.G., LaVigna C., Polyakov V., "Tools for Integrated Modeling, Design, and Nonlinear Control", *IEEE Control Systems*, vol 15 n° 2, pp 65-77, 1995.
- [Bohn and Atherton , 1995], "An Analysis Package Comparing PID Anti-Windup Strategies", *IEEE Control Systems*, vol 15 n° 2, pp 34-40, 1995.
- [Bobrow, 1984], "Qualitative reasoning about Physical Systems : An Introduction", *Artificial Intelligence*, 24, pp : 1-5, 1984.
- [Cassar and Staroswiecki, 1996], "Pour una approche unifiée de la surveillance" in *Preprints of Surveillance des systèmes continus*, Tome 1, Ecole d'Été d'Automatique de Grnoble, Sept. 1996.
- [Cheung and G. Stephanopoulos, 1990], "Representation of process trends - Part I and Part II", *Computer Chemical Engineering*, 14, pp 495-540, 1990.
- [Chipperfield and Fleming, 1995], "PARSIM: A parallel Optimization Tool", *IEEE Control Systems*, vol 15 n° 2, pp 48-53, 1995.
- [Colomer et al. 1996], Colomer J., C. Pous, J. Melendez, J. Ll de la Rosa, J. Aguilar, "Abstracting Qualitative Information for process Supervision", *IEEE International Symposium on CACSD*, pp. 410-415, Dearborn, MI, 1996.

- [Colomer et al. 1997], Colomer J, Melendez, J, De la Rosa J.L., Aguilar, J. "A qualitative/quantitative representation of signals for supervision of continuous systems", in ECC97, ECC382.pdf, Brussels, 1997.
- [De Kleer and Brown, 1984] "A qualitative physics based on confluences", *Artificial Intelligence*, N° 24, pp 7-83, 1984.
- [De la Rosa J.Ll, 1994], "Heuristics for cooperation of expert systems. Application to process control", Universitat de Girona, 1994.
- [De la Rosa et al., 1995], De la Rosa J.Ll., Colomer, J., Meléndez, J., "Qualitative Modelling for partially known biotechnological process", in '*Current trends In Qualitative Reasoning and Applications*', pp. 114-119, Monograph CIMNE N° 33, Barcelona (Spain), 1995.
- [Denoeux, Masson and Debuissou, 1996] "System diagnosis using Pattern Recognition Techniques : A Survey", in *Surveillance des systèmes continus, Tome 1, Ecole d'Eté d'Automatique de Grenoble*, Sept, 1996.
- [Dorf R.C., 1993], "Exploring Control Design Variables", *Proceedings of the American Control Conference*, pp 3062-3066, 1993.
- [Du, Elbastawi and Wu, 1995a] R. Du, M.A. Elbastawi and S. M. Wu, "Automated process monitoring., Part I : Monitoring methods", in *Journal of Engineering for Industry*, 117 : 121-132, 1995
- [Du, Elbastawi and Wu, 1995b] R. Du, M.A. Elbastawi and S. M. Wu, "Automated process monitoring., Part II : Applications", in *Journal of Engineering for Industry*, 117 : 133-141, 1995
- [Frank, 1996], "Analytical and Qualitative Model-based Fault Diagnosis -A Survey and Some New Results", in *European Journal of Control* ,2 :6-28, 1996.
- [Frank and Köppen-Seliger, 1995] "New developments using AI in fault diagnosis", in IFAC/IMACS International Workshop, Bled Slovenia, Dec. 1995.
- [Ganz, Kolb and Rickli, 1993], "A Data Management Tool for Computed Aided Control Engineering", *Proceedings of the American Control Conference*, pp 3076-3080, 1993.
- [Gentil, 1996], "Intelligence Artificielle pour la Surveillance des procédés continus", in *Surveillance des systèmes continus, Tome 1, Ecole d'Eté d'Automatique de Grenoble*, Sept, 1996.
- [Grübel, 1993], "ANDECS and CACE 'Frameworks Reference Models ", *OPEN-CACSD : Newsletter of the IFAC/IEEE-CSS Working Group Guidelines for Open CACSD Software*, Swansea, 1 (11), February, 1993
- [Grübel, 1995], "The ANDECS CACE Framework", *IEEE Control Systems*, vol 15 n° 2, pp 8-13, 1995.

- [IEEE Control Systems, 1995], "Special Issue on CACSD", *IEEE Control Systems*, Vol 15, N2, pp6-85, April 1995.
- [Isermann and Ballé, 1996], "Trends in the application of model based fault detection and diagnosis of technical processes", in *IFAC -13th Triennial World Congress*, ref. 7f-01 1, San Francisco, USA.
- [Jacobstein and Kitzmiller, 1988], "Integrating Symbolic and Numerical Methods in Knowledge-Based Systems : Current Status, Future Prospects, Driving Events", in "Coupling symbolic and numerical computing in expert systems, II", JS Kowalik and CT Kitzmiller Edt., Elsevier Science Publishers B.V., North Holland, 1988.
- [James et al., 1995], "The state of Computer-Aided Control System Design", *IEEE Control Systems*, vol 15 n° 2, pp 6-7, 1995.
- [James, 1988], "Lessons learned in coordinating symbolic and numeric computing in knowledge-based systems for control design", *Coupling Symbolic and Numerical Computing in Expert Systems II*, J.S. Kowalik and C.T.Kitzmiller Edts., Elsevier Science Publishers B.V., 1988.
- [Jobling et al. 1994], Jobling C.P., Grant P.W., Barker H.A., Townsed P, "Object-oriented Programming in Control System Design: a Survey", *Automatica*, vol 30, n° 8, pp 1221-1261, 1994.
- [Koch, 1993], "Modular Reasoning. A new approach towards intelligent control", *Doctoral thesis Swiss Federal Institute of technology*, Zurich, 1993.
- [Kuipers, 1986], "Qualitative simulation", *Artificial Intelligence*, Vol 29, pp289-338, 1986.
- [Lee et al. , 1993], LeeT.H., C.C. Hnag, W.K. Ho, P.K. Yue, "Implementation of a knowledge-based PID Auto-tuner", *Automatica*, Vol. 29, No 4, pp 1107-1113, 1993.
- [Lynch and De Paso, 1992], "An Object Oriented Intelligent Control Architecture", *American Control Conference*, 1992.
- [Maciejowski and Szymkat, 1994], "Containers - a step towards objects with Matlab", *Proc. IFAC/IEEE Joint Symp. on Computer Aided Control System Design*, Tucson, AZ, March, 1994.
- [Mallat, 1989], Mallat S.G. «A theory for multiresolution signal descomposition: the wavelet transform», *IEEE Transactions on pattern analysis and machine intelligence*, vol 11, n° 7,pp 674-693, 1989.
- [Maquin and Ragot, 1996], "Méthodes de base de la surveillance: systèmes statiques et dynamiques", in *Surveillance des systèmes continus, Tome 1, Ecol d'Eté d'Automatique de Grenoble*, Sept., 1996.

- [Martinez, 1997], Martinez Arteaga, A., "Exemple d'aplicació del sistema expert CEES a una planta industrial en temps real", PFC-EPS, Universitat de Girona, 1997.
- [Matlab 1996], Matlab 5.0 Using Matlab, The MathWorks, Inc, Dec 1996.
- [Melendez et al. 1995], Melendez, J., Colomer J. and De la Rosa J.Ll., "Linking G2 and Matlab for Developing an Expert Diagnostic System", in '*Supervision de processus à l'aide du système expert G2TM*', pp. 91-110, Ed. Hermès, Paris, 1995.
- [Melendez et al. 1996a] Melendez J., Colomer J., De la Rosa J.Ll., Vehí J., "Dealing with Qualitative Information in Simulation for Supervisory Systems design", in *Modelling and Simulation, ESM'96*, June, Budapest, 1996.
- [Melendez et al., 1996b], Melendez J., J. Colomer, J. Ll De la Rosa, J. Aguilar and J. Vehi, "Embedding Objects into Matlab/Simulink for Process Supervision", *IEEE International Symposium on CACSD*, pp.20-25, Dearborn, MI, 1996.
- [Melendez et al., 1996c], Melendez J., J.LL. de la Rosa, J. Colomer, J. Vehí i C. Pous, "Incrustación de objetos en Simulink. Integración de herramientas de ayuda al diseño de estructuras de supervisión" , *II Congreso de usuarios de MATLAB (MATLAB'96)*", pp. 423-429, Madrid, Sept. 1996.
- [Melendez J., 1996d] "Integrating tools for computer aided expert supervision design", *Ir seminari de treball en automàtica robòtica i percepció, Edicions UPC*, pp: 29-43, Barcelona, 1996.
- [Melendez et al., 1997a], Melendez J., J.Ll. de la Rosa, J, Colomer, J. Aguilar-Martin, O. Contreras "A Framework for Dealing with Significant Information and Knowledge Representation in Expert Supervisory Systems Design" *7th IFAC Symposium on Computer Aided Control System Design (CACSD'97)*, pp :397-402. Gent (Belgium) April 28-30, 1997.
- [Melendez, et al., 1997b], Melendez J., Colomer J., De la Rosa J Ll, Waissman-Vilanova J, Aguilar-Martin J., "Supervisory diagnostic structure using qualitative signal abstraction and rule-based inferential reasoning : A two tank illustrative example" , *COMADEM '97*, Helsinki (Finland), 1997, June 9-11.
- [Melendez, 1998], Melendez J., Colomer J., De la Rosa J.Ll., Contreras O., "Expert diagnostic using qualitative data and Rule-based inferential reasoning" , submitted at the *11th, International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA'98*, Castellon de la Plana, June, 1998.

- [Milot, 1996] “ De la Surveillance à la Supervision : l’integration des Operateurs Humains”, in *Surveillance des systèmes continus, Tome 1, Ecole d’Eté d’Automatique de Grenoble*, Sept, 1996.
- [Milne, 1987], “Strategies for diagnosis”, *IEEE transactions on Systems, Man and Cybernetics, SMC-17*, pp. 333-339, 1987.
- [Munro, 1990], “ECSTASY-A Control System CAD Environment”, in Proc. *11th IFAC World Congress on Automatic Control*, Tallin, Estonia, 13-17 August, 1990.
- [Nebendahl, 1988] Nebendahl Dieter, “Expert Systems. Introduction to the Technology and Applications”, *Siemens Aktiengesellschaft, John Wiley and Sons Limited*, 1988.
- [Ogunnaiké, 1995], Ogunnaiké, B.A.,”The Role of CACSD in Contemporary Industrial Process Control”, *IEEE Control Systems*, vol 15 n° 2 ,pp 41-47, 1995.
- [OMG, 1995], “The Common Object Request Broker : Architecture and Specification ”, 2 Ed., Object Management Group, July 1995.
- [Ong, 1992], "Autonomous Control System Design", *ACC/TM3*, p.p. 1898-1899, 1992.
- [Pal, 1995], Kajnal Miklos Pal, “Model Based Fault Detection”, *Contribution to TEMPUS project MODIFY*, Duisburg, 1995.
- [Piera, 1995], Piera N., “Current trends In Qualitative Reasoning and Application”, pp. 114-119, *Monograph CIMNE N° 33*, Piera Ed., Barcelona (Spain), 1995.
- [Rakoto-Ravalontsalama N., 1993], "Sur l’interface numerique-symbolique dans un schema de supervision de systemes dynamiques a l’aide de systemes experts", *These de Doctorat Universite Paul Sabatier*, Toulouse, France, 1993.
- [Rengasamy, 1995], “A framework for integrating process monitoring, diagnosis and supervisory control”, *PhD. Thesis of the Purdue University*, August, 1995
- [Rimvall et al., 1994], Rimvall C.M, Farrell J.A, Radecki M, Idelchick M, “An open architecture for Automatic Code Generation using the BEACON CACE environment”, *Joint IEEE/IFAC Symposium on Computer-Aided Control System Design*, Tucson, AZ; March 1994. And also http://www.phrantic.com/j_alan/hitech/case/beacon.html (Sept. 1997).
- [Rutz R. and Richert, 1995], "CAMEl: An open CACSD environment", *IEEE Control Systems*, vol 15 n° 2,pp 26-33, 1995.

- [Sàbat, 1996], ‘Supervisió experta mitjançant el sistema expert CEES. Exemple d’utilització’, *Proyecto Fin de Carrera, ETET, Vilanova i la Geltru*, UPC, 1996.
- [Sarrate R. et al., 1995], "Generación de eventos por análisis de datos basado en ventanas deslizantes", *XVI Jornadas de Automática*, Donostia, Spain, 1995.
- [Saifuddin, 1996] Saifuddin, Anita Bilqees, “Computing Environments for Control Engineering”, *Ph. D. Thesis at the University of Cambridge*, UK, March, 1996.
- [Saifuddin et al., 1996], Saifuffin A.B., Maciejowski J.M. and Szymkat M., (1996), "Computational chains for CACSD Using Matlab Containers", *Proceedings of the 1996 IEEE international Symposium on CACSD*, pp. 392-397, Dearborn. MI, 1996.
- [Shen and Leitch, 1993], “Fuzzy Qualitative simulation”, *IEEE, Trans. on Systems Man and Cybernetics*, V. 23, N4, pp :1038-1064, 1993.
- [Simulink 1993], Simulink 1.3 User's Guide, The MathWorks, Inc, April 1993.
- [Simulink 1994], Simulink 1.3 Release Notes, The MathWorks, Inc, May 1994.
- [Simulink 1994], Simulink 1.3 Release Notes, The MathWorks, Inc, May 1994.
- [Simulink 1996], Simulink 2.0 Using Simulink, The MathWorks, Inc, Dec 1996.
- [Taylor et al., 1990], Taylor J.H., Frederick D.K., Rinvall C.M., Sutherland H.A., “A computer aided control engineering environment with expert aiding and data-base management”, *IFAC World Congress*, Tallim USSR, 13-17, Aug. 1990.
- [Taylor et al., 1993], Taylor J.H., Rinvall, C.M., Sutherland, H.A. “Computer-Aided Control Engineering Environments”, *Chapter 17, CAD for Control Systems*, D.A. Linkens Ed. Marcel Dekker Inc. 1993.
- [Vinoski, 1997], “CORBA : Integrating Diverse Applications Within Distributed Heterogeneous Environments”, *Communications Magazine*, Vol.14, No.2, February, 1997.
- [WGS Newsletter, 1997], Working Group on Software, n° 11, <http://www.win.tue.nl/wgs/NEWSLETTER/NEWSL11/wgsnews11.html>, January, 1997.

