A Model-Based Control System Concept

Årzén, Karl-Erik

1992

Document Version:
Publisher's PDF, also known as Version of record

ISSN 0280–5316
ISRN: LUTFD2/TFRT--3213--SE

A Model-Based Control System Concept

Karl-Erik Årzén

Department of Automatic Control
Lund Institute of Technology
December 1992

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
December 1992

Document Number
ISRN LUTFD2/TFRT--3213--SE

Author(s)
Karl-Erik Årzén

Supervisor

Sponsoring organisation
IT4, NUTEK

Title and subtitle
A Model-Based Control System Concept

Abstract

This paper presents an overview of a new concept for DCSs developed within the KBRTCS (Knowledge-Based Real-Time Control Systems) project performed between 1988 and 1991 as a part of the Swedish IT4 Programme. The partners of the project have been the Department of Automatic Control at Lund Institute of Technology, Asea Brown Boveri, and during parts of the project, SattControl, and TeleLogic.

The aim of the project has been to develop a concept for future generations of DCSs based on a plant database containing a description of the plant together with the control system. The database is object-based and supports multiple views of an objects. A demonstrator is presented where a DCS system of this type is emulated. The demonstrator contains a number of control, monitoring, and diagnosis applications that execute in real time against a simulation of a Steritherm sterilization process.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
30

Recipient's notes

Security classification

A Model-Based Control System Concept

Karl-Erik Årzén
Department of Automatic Control
Lund Institute of Technology
Box 118, 221 00 Lund, Sweden

Abstract: A new concept for distributed control systems is outlined. The concept is based on a plant database containing descriptions of the plant together with the control system. The database is object-based and supports multiple view of an objects. A demonstrator is presented where a DCS system of this type is emulated. The demonstrator contains a number of control, monitoring, and diagnosis applications that execute in real time against a simulation of a Steritherm sterilization process.

1. Introduction

The evolution of industrial control technology during the last 30 years has moved in the direction of computer-based implementation and integration of functionality (Lukas, 1986). Analogue controllers have been replaced by digital controllers. Discrete controllers are replaced with distributed control systems (DCSs) that also includes the functionality of relay systems and programmable logic controllers (PLCs). Supervisory control and monitoring is integrated with the low level continuous control loops, interlocking logic, and sequence logic. On the higher levels of the DCS, information management systems provide plant-wide services such as planning, scheduling, and optimization.

The process operation tasks, i.e., control, monitoring, fault detection, diagnosis, scheduling, planning, production optimization, etc., interact strongly with each other and should not be treated as individual problems (Pekny *et al*, 1991). The interaction may involve exchange of information between the tasks or the fact that most of the tasks are based on some, implicit or explicit, description or model of the process. However, all too often the tasks are treated individually resulting in poor process operation. There are two major reasons for this. The first is the lack of unified theoretical frameworks and design methodologies for integrated process operation. An example of a problem that remains to be solved are how control and on-line diagnosis should be combined. The second problem is the lack of integrated environments where the operation tasks can be designed, and implemented. The latter problem is the topic of this paper.

Current DCSs focus almost entirely on process operation. However, there is a clear need to integrate also other plant activities such as plant design, control system design and configuration, plant maintenance, and re-design. For example, the process knowledge, e.g., dynamic models, derived during plant design is only marginally accessible during plant operation. The heuristic plant knowledge obtained from experience by operators is only seldomly fed back to the plant designers.

A result of the lack of plant integration is that process knowledge is spread out, duplicated, and documented in a variety of different forms and places, e.g., in CAD systems, in various documents and manuals, in plant engineering and design databases, in the models of process simulators, in the knowledge base of real-time expert systems, and in the DCS database. To retain consistency among all this information is very difficult. An example of the strive towards integrated representation of plant information is the CALS (Computer-aided Acquisition and Logistic Support) project initiated by the US Department of Defense (Smith, 1990). The ultimate goal of CALS is to develop integrated product databases that will contain the information needed for the design, manufacturing, and maintenance of complex industrial systems with the primary applications in the weapons industry.

Plant integration has hitherto mainly been adopted by the manufacturing industry where, e.g., CIM has become a well-known acronym. However as competition increases integration will also be necessary in the process industry. For example, due to economic constraints and safety regulations future chemical plants will be more closely coupled with fewer intermediary storages. This puts higher demands on integrated control and supervision in order not to jeopardize safety. Increased quality considerations according to, e.g., ISO 9000 also increases the demand for integration. Process industry, in general, also becomes more customer oriented with requirements on production of smaller volume, high valued added products. The CIM type integrated manufacturing is often named CAPE (Computer-Aided Plant Engineering) or CAPO (Computer-Aided Process Operation) in the process industry (Reklaitis and Spriggs, 1987).

This paper presents an overview of a new concept for DCSs developed within the KBRTCS (Knowledge-Based Real-Time Control Systems) project (Årzén, 1990; Årzén *et al*, 1990) performed between 1988 and 1991 as a part of the Swedish National Information Technology Research Programme, IT4. The partners of the project have been the Department of Automatic Control at Lund Institute of Technology, Asea Brown Boveri, and during parts of the project, SattControl, and TeleLogic. The aim of the project has been to develop a concept for future generations of DCSs based on a plant database containing a description of the plant together with the control system. This plant database, in the sequel referred to as the knowledge base, constitutes a single representation for "all" the information needed for the different phases of a plant life cycle, e.g., design, operation, maintenance, etc. The background and further motivation for the project is given in Section 2.

The project has focussed on the structuring concepts needed in the knowledge base. The knowledge base is essentially object-oriented with classes, objects (i.e., instances), and hierarchical objects. The knowledge base also contains objects representing systems by which is meant flows of mass, energy, or information in the process. The knowledge base supports multiple representations, or views, of a single physical object. In order to hold together different views of the same object the concept of **multi-view objects** has been defined. The control system concept is overviewed in Section 3.

Steritherm, a sterilization process from Alfa-Laval has been used as a test process in order to evaluate the concept on a "real" process that is representative for a large class of industrial processes. As a means for visualizing the concept a demonstrator has been developed. The demonstrator implemented in G2 from Gensym Corp (Moore *et al*, 1990) contains the knowledge

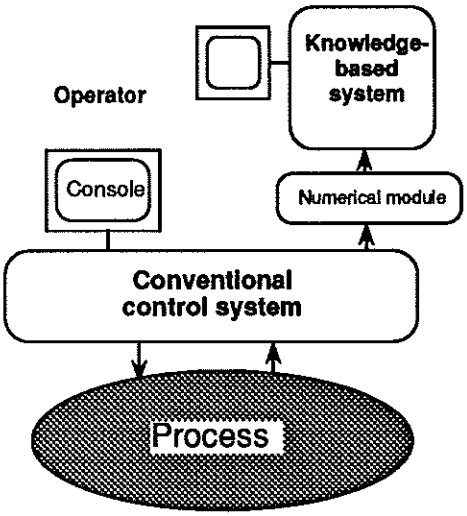

Figure 1. Add-on solution for real-time knowledge-based systems

base containing the plant and control system description, an operator interface, a process engineer interface, and a graphical browser for inspecting the knowledge base. Within the knowledge base different control, monitoring, diagnosis, and planning applications have been implemented. These include Grafcet-style sequential function charts for the sequence control logic, rule-based monitoring, fault-tree based off-line diagnosis, on-line diagnosis based on quantitative constraint models, on-line diagnosis based on qualitative signed directed graph models, alarm analysis based on functional models, and order scheduling. The demonstrator is described in Section 4.

2. Background and Motivation

When the KBRTCS project started in 1988 its main focus was real-time knowledge-based systems. For a long time there has been a strong interest in knowledge-based systems (KBSs) within the process industry. Many applications of different types including expert control, fuzzy control, on-line monitoring and diagnosis, off-line trouble-shooting, and planning, have been proposed and tested in pilot projects. Systems that are in regular, day-to-day operation are still not common. The major reason for this is the lack of integration between KBSs and conventional control system (Årzén, 1989; 1991a).

The interest in the process industry for knowledge-based systems has created a market for expert system tools specially aimed at real-time, on-line applications. Today several such tools exist with G2 as a good example. Usually, these tools are separate, "add-on" systems that have an on-line interface to the control system as shown in Fig. 1. Through the interface, process measurements and events are transferred to the expert system for analysis. The output from the expert system is advice to the user, usually the process operator, and parameter changes to the control system.

Although, in many cases the only possible solution, the "add-on" solution, has many problems. The major problems stem from the fact that the systems are different. The systems come from different suppliers, require different expertise, use different hardware and software and above all, in many cases have different end-user interfaces. The interface between the KBS and the

control system is separate from the control systems internal communication network and may cause communication bottlenecks. Furthermore, large amounts of data and information such as process parameters and schematics are represented redundantly in both systems causing problems with maintaining consistency.

With this as a background the goal of the project was initially to specify a new concept for DCSs that allowed the merging of the functionality of current DCS systems with the functionality of real-time expert system tools. This new DCS was termed a *knowledge-based control system (KBCS).* The aim of the concept was to combine the strong features of real-time expert system tools such as explicit knowledge representation, support for representing heuristic knowledge, object orientation and support for rule-based reasoning, support for temporal reasoning and for reasoning about dynamic environments, and user-friendly interfaces, with the strong features of modern distributed control systems such as support for algorithmic representation of control logics, speed, distribution, good graphical operator interfaces, and hardware and software reliability. The time horizon before DCSs of this type would emerge was estimated to be 10–15 years.

However, gradually the focus of the project was widened to also include integration of other types of information and knowledge than the traditional expert system type "heuristic knowledge" and to look upon other techniques than knowledge-based. Hence, in retrospect the real focus of the project has been to find a system architecture that supports the representation and real-time manipulation of a large variety of different types of information concerning the

- design of the plant and the control system,
- process components, raw materials, and products,
- plant documentation,
- simulation of the plant,
- plant operation, e.g., continuous control, sequence control, monitoring, on-line diagnosis, etc.,
- maintenance, and the
- long-term planning.

Hence, the project aims to cover so disparate items such as on-line information and documentation systems, 3-D geographical process descriptions, process models that could have different usages, e.g., simulation, diagnosis, etc., and on different forms, e.g., quantitative models, qualitative models, functional models, etc., control and monitoring algorithms, heuristics for process operation and monitoring, etc.

The main reason for the widened scope of the project was the insight that the problems with redundant information stored in more than one place with accompanying problems of consistency are general problems in process design, operation, and maintenance. Another common factor is the use of models. Models of various forms, e.g., quantitative, qualitative, static, dynamic, topological, functional, behavioral, etc., and of various degrees of detail are used, explicitly or implicitly, in almost all phases of a plant's life

cycle. Models are used off-line as a means for designing the plant and the control system as well as on-line as a part of the controllers and the diagnosis systems. Ideally one would like to have one "high-fidelity" model from which all other models could be automatically generated. This is, however, hardly realistic. Therefore the knowledge base must support multiple models of the plant, each intended for a specific task. In the AI community a similar situations exist for knowledge representation. A single uniform all-purpose knowledge representation technique does not exist. Knowledge representation and use cannot be completely separated.

The main part of the project has focussed on the internal structure of the knowledge base. With the large amount of information of different kind that should be represented in the knowledge the right choice of structuring mechanisms is essential. The structuring mechanisms form the backbone of the knowledge base onto which the various pieces of information can be attached.

Maintaining the knowledge base consistent is important. Automatic methods for ensuring consistency in knowledge bases of the kind envisioned in this project do not exist. The approach to support consistency is instead to store all information regarding an entity in one place in the knowledge base and thereby simplify for the users to keep the information consistent.

3. The Concept

The KBCS concept is based on a knowledge base that is shared between all the users of the systems. The knowledge base should have the potential to represent what today is found in

- conventional DCS systems, e.g., descriptions of control logic, alarm logic, operator interfaces, real-time and historic data;
- CAD systems, e.g., electrical drawings, 3-D object descriptions;
- information management systems, e.g., descriptions of higher level supervisory and planning functions;
- on-line documentation systems, e.g., various manuals, data sheets;
- design databases, e.g., process component information, product information;
- plant simulators, e.g., dynamic models; and
- real-time knowledge based systems, e.g., qualitative models and heuristics.

Since DCS systems are distributed it is clear that the knowledge base cannot be one physical unit. Instead it must be distributed onto multiple nodes of the DCS. For example, due to real-time requirements the dynamically updated parts of the knowledge base, e.g., sensor signals must reside at the place where they are most frequently used. It is also clear that different implementation techniques, e.g., real-time databases, hypertext databases, relational data bases, object-oriented databases, etc., must be used to implement the various part of the knowledge base. However, what is important is

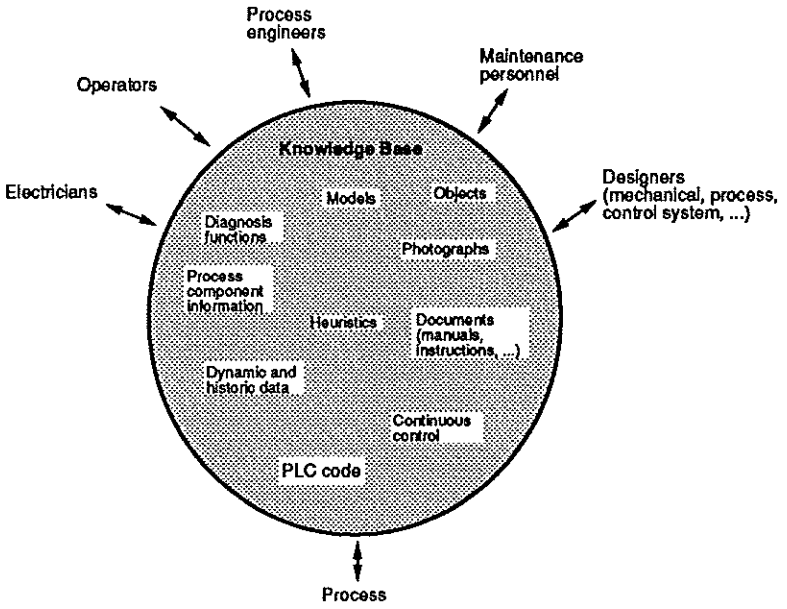

Figure 2. The knowledge base

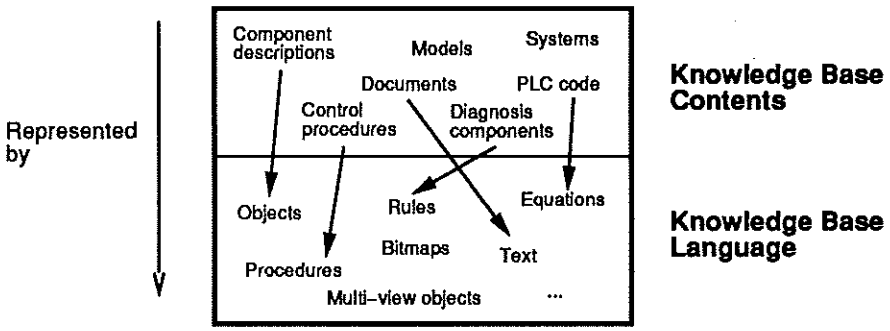

Figure 3. Knowledge base contents and language

that the knowledge base appears to the users as a single, uniform database according to Fig. 2.

Some parts of the knowledge base are executable, i.e., they consist of control code, logic, etc., that should be executed on-line. Other parts are of a more static nature, e.g., documents. The contents of the knowledge base are represented by a knowledge base language that contains objects, multi-view objects, rules, equations, procedures, text, bitmaps, etc., as built in data and program structures, see Fig. 3.

Two sources of influence for the knowledge base language have been SattLine from SattControl (Elmquist, 1991) and G2. Sattline is a DCS from SattControl that is based on an object-based language that contains control logic descriptions together with the operator interface description. Sattline is programmed from the operator stations and the executable code is being distributed out to the control units. G2 is based on a hybrid programming language that combines objects, rules, procedures, and equations.

User interfaces: Each user of the KBCS interacts with the system via his **custom user interface**. These interfaces are designed to meet the requirements from different users regarding what information that should be pre-

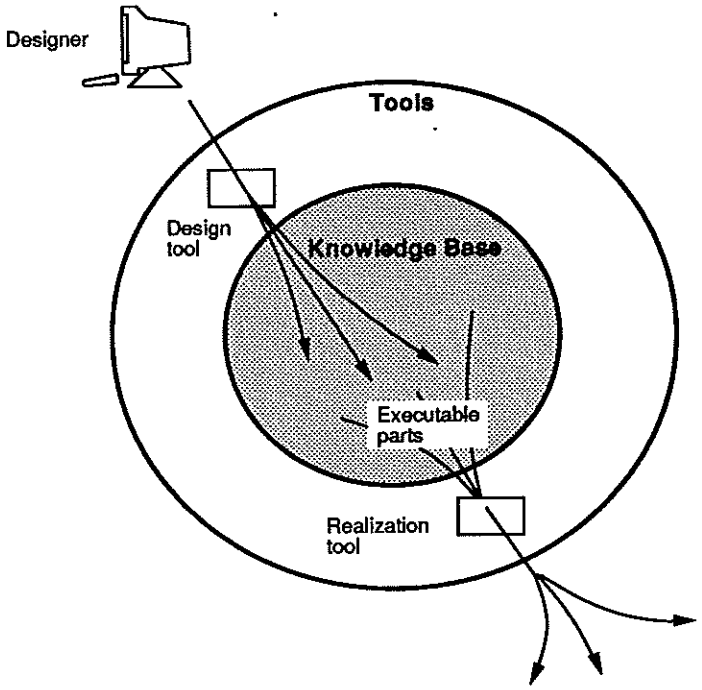

Figure 4. Tools and knowledge base

sented and how it should be presented. Through the custom interface a user has access to the parts of the knowledge base that are needed for him to fulfill his normal tasks. Personal preferences on how the information should be presented, e.g., in terms of picture layout, types of interaction facilities, etc., are also reflected in the custom user interface. The definitions of default versions of the custom user interfaces for the different user groups are a part of the knowledge base.

Knowledge base browser: In addition to the custom interfaces all users also have access to the **knowledge base browser.** The browser constitutes a common, user-independent graphical interface through which the entire contents of the knowledge base is made available to the users. The graphical presentation of the knowledge base contents, i.e., the objects and their connections, is the same for all users when seen through the browser. This presentation format is a default format that constitutes a common reference frame for all users. In the custom interfaces, however, the same knowledge base object may be presented differently to different users.

Tools: The knowledge base is accessed through tools. The tools are of two main types: **design tools** and **realization tools.** Design tools are used to support the different designers when they build up the knowledge base. The design tools are either application independent or specific to an application, e.g., to a specific diagnosis methodology. The design tools are tightly connected to the custom user interfaces for the designers using the design tools. Realization tools extracts the executable elements of the knowledge base language, compiles them to a form better suited for efficient execution, and distributes them to the processors where they should execute. The realization tools also set up the communication links between different processors and between the processors and the user interfaces. The tools and knowledge base architecture is shown in Fig. 4.

The tools could also include, e.g., simulators that operates on simulation equations in the knowledge base or optimization routines. There is a trade-off between whether a task should be represented within the knowledge base itself, i.e., expressed in terms of the knowledge base language or if it should be implemented by a tool that operates on the knowledge base.

Structuring concepts

The structuring concepts used with the knowledge base are objects, hierarchical objects, systems, and views.

Objects: Objects are the basic and most general knowledge representation formalism used in the concept. Object-oriented representation and programming have proven to be very powerful paradigms. A major strength of objects is that they represent a concept with its associated characteristics as a single entity. In the project the usual meaning of object-oriented programming is used with class definitions, inheritance of attributes, and methods.

Objects may represent physical entities, e.g., process components, as well as abstract entities. It is natural to also view the built-in data and program structures in the knowledge base language as objects.

An object may have an associated graphical representation, i.e., an icon. This icon is used to represent the object when the object is seen from the knowledge base browser. Objects have relations to other objects. The relations may also have graphical representation. i.e., they are drawn as connections between the objects. Connections may represent physical connections, e.g., pipes or wires, or abstract relations. Using connected objects various types of graphical structures can be represented, e.g., process schematics, electrical circuit drawings, graphs, etc.

A problem with object-oriented structuring is that it is highly application dependent. For a given problem, e.g., diagnosis or simulation, it is quite easy to find classes and objects that naturally match the problem at hand. However, another application, e.g., planning, dealing with the same physical process might require another, quite different, class and object structure. A situation arises where the users need to represent the same physical entity, e.g., a process component, as several objects in the knowledge base, each with its own class definition, attributes, graphical icon, and connections and relationships to other objects. This situation conflicts with the ambition that the same knowledge should only be represented once in the knowledge base, and all knowledge pertaining to the same object should be maintained as one localized unit.

Hierarchical objects: A hierarchical, or composite, object is an object that has an internal structure of, usually interconnected, objects representing the subparts of the object. The composite object has an icon and may be connected to other objects. When, however, the composite object is zoomed in upon or "opened" the internal structure of the object is shown according to Fig. 5.

Hierarchical decomposition is quite natural in a plant database. A plant consists of processes. A process consists of subprocesses or unit processes. A unit process consists of process components, etc. However, in the same way that the class structure is application dependent, the decomposition

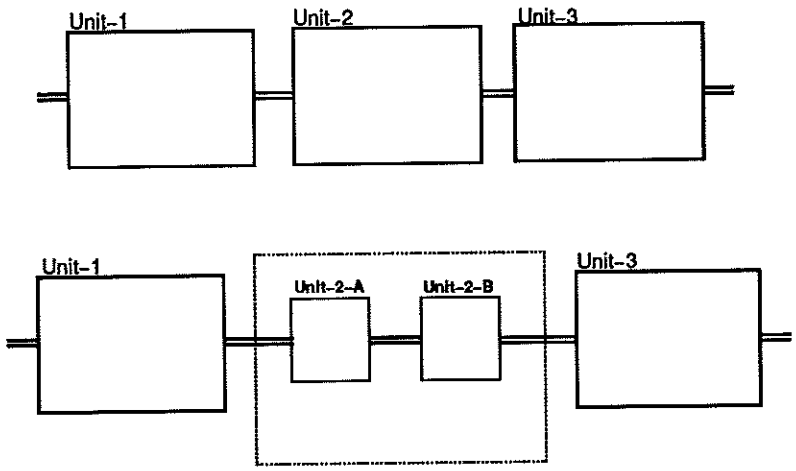

Figure 5. Unit-2 is a composite object that internally consists of Unit-2-A and Unit-2-B.

hierarchy is also application dependent. A single hierarchical decomposition of a plant does not exist. Instead, the purpose for which the knowledge should be used determines the appropriate hierarchy levels.

Systems: A continuous process can be divided into a number of systems. The systems correspond to the different flows of material, energy, or information in the process. The most important system is the main system. In a pulp process this would correspond to the pulp system that describes how wood chips are converted to pulp, flowing through the process from the impregnation vessel, through the continuous digester, oxygen bleacher, diffuser bleacher, etc. In the Steritherm case the main system is the milk system.

The main product system is the raison d'être for the process. In order for this system to function properly a number of support systems must exist. Examples of support systems are the electrical system, the steam system, the pneumatic system, the hydraulic system, the control system, etc. The control system has a special role. The control system in the knowledge base constitutes a description of the actual control system that controls and monitors the process.

Structuring plant knowledge according to systems has many advantages. It well reflects the situation of today in the process industry. Different user groups work with different systems in the process. For example, the main system, the electrical systems, and the control system are the responsibility of the operators and process engineers, electricians, and the instrument engineers, respectively. Furthermore, different systems are documented separately.

A system is naturally represented by a composite object. The internal structure of the object consists of process, subprocesses, or process components that are a part of this system. It is unusual that, e.g., a process component is a part of only one system. For example, a pump may be a part of the main system. However, the pump has a power supply and is therefore also a part of the electrical system. Furthermore the pump may be water-cooled and, hence, a part of the cooling water system. In each of the systems that a component is a part of it, it may have a different graphical representation, different attributes, and different relations with other objects. The situation is shown in Fig. 6.

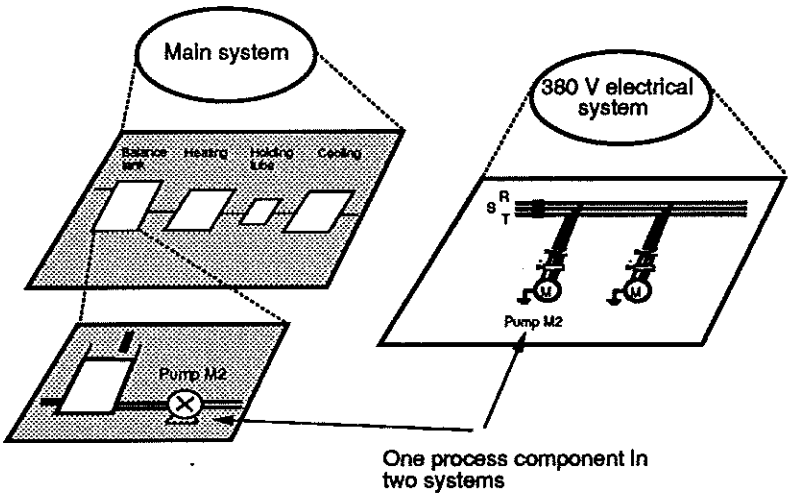

Figure 6. A component that is a part of multiple systems

Views: A system oriented plant decomposition structures the knowledge base after the physical systems of the process. An object that appears in multiple systems needs to be described differently in each system as already pointed out. This is, however, not enough. It is often necessary to be able to describe an object appearing in one system in more than one way at the same time. It is also necessary to describe an entire system in more than one way at the same time.

The different ways of describing an object are named the views of the object. Each view represents the object in a particular context. Describing an object from several views is a way of structuring the knowledge about an object into "natural" parts, i.e., to structure the representation of knowledge about the object. In software terminology, the term view can also be used in connection with how information is presented to different users. In the concept the latter meaning of view is named **user view**. User views are what the different users see of the knowledge base through their custom user interfaces. There may be a one-to-one correspondence between the views in the knowledge base and the user views but it is not necessary.

A very simple example of when an object needs to be represented from different views is a electrical circuit consisting of a battery and a lamp. This is represented by an electrical system object. The electrical system object can, e.g., be described in terms of its internal structure as described by an electrical circuit schematic. This consists a topological view of the object. It may also be of interest to have a functional view of the circuit that leaves out implementation details. Furthermore, one may want to describe the object geographically in terms of its size, shapes, coordinates, etc. The situation is shown in Fig. 7. The electric circuit is represented by three views: a geographical view, a topological view, and a functional view. The object representing the battery is a part of all these three views. Therefore, the battery also has three views and in each of these it can have different attributes.

Which views that are used to a large degree depend on the application. However, the topological, functional, and geographical views are of a more general nature and probably apply to processes in general, independently of the application. One should have in mind that these views are not always applicable, and that others always exist. It should also be remembered that

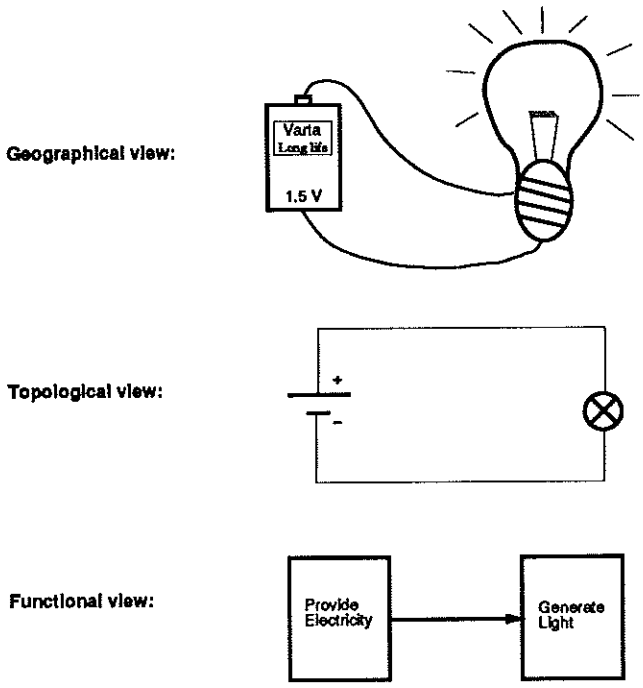

Figure 7. Different views of a simple electrical system.

all views do not occur on all levels in the plant hierarchy and that in a particular application perhaps only a very small number of views are used.

An example of other views is shown in Fig. 8. Here, a digraph view is used to describe the causal relationships among process variable in a system and a constraint equation view is used to describe the governing equation constraints in the system. Both the digraph view and the constraint view could be the basis for different model-based diagnosis schemes as will be shown in Section 4. Note here that the sensor objects appear in the topological view. They also appear as nodes in the digraph view and as failure assumptions in the constraint view.

Multi-view objects

The problem with objects that need to be represented in more than one way at the same time is solved using multi-view objects. A multi-view object can be seen as a meta-object that groups together all the representations, i.e., views, of an object. Each view is represented by an "ordinary" object, i.e, it is defined in a class definition, has attributes, may be composite, has relations to other objects, etc. The relationship between multi-view objects and "ordinary" objects is shown in Fig. 9.

Multi-view objects are used to represent everything that needs more than one representation in the knowledge base. For example, systems are represented by multi-view objects. The views of the multi-view object represent the views of the system, i.e., the topological view, the functional view, etc. Process components that are part of more than one system or of more than one view of the same system is represented by multi-view objects. Each view of the multi-view object represents the process component in a particular context.

Multi-view objects can also be composite. For example, the multi-view object representing the entire plant is composed of the multi-view objects

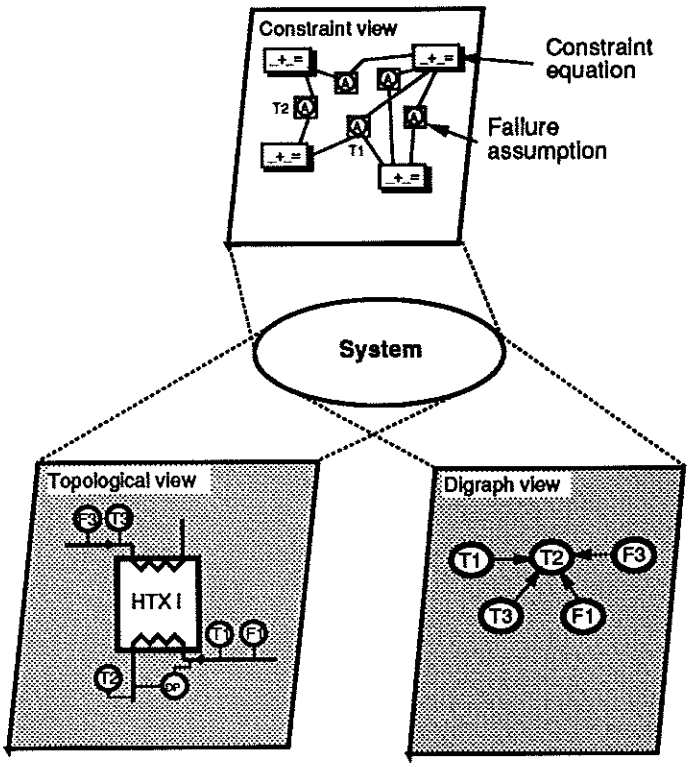

Figure 8. Digraph and constraint views

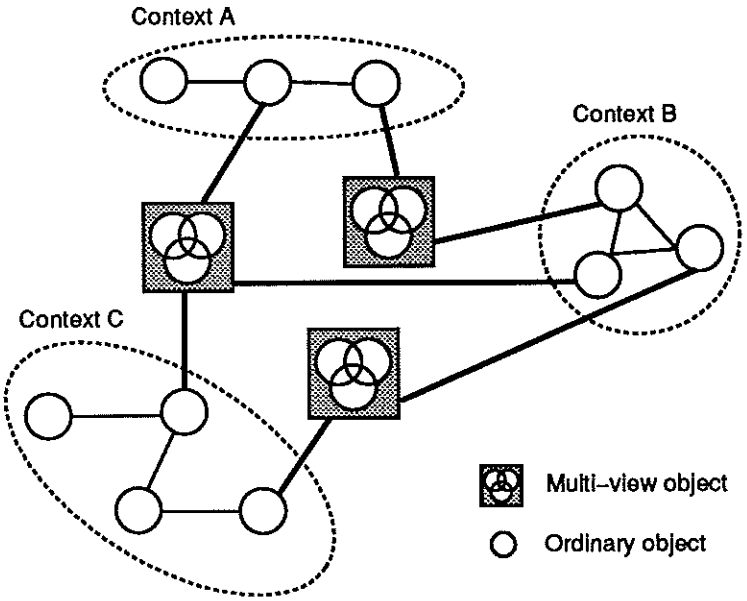

Figure 9. The relationship between ordinary objects and multi-view objects

representing the different systems in the plant.

Multi-view objects are defined in multi-view class definitions. The multi-view class definition defines the views of the multi-view object. The definition includes the names of the views and what classes these views should be instances of. A multi-view class can be derived from another multi-view class. In that case the class inherits the views of its superior class.

The multi-view class definition also defines which attributes that should be shared between the views. This is performed through an attribute equivalence declaration. The attribute equivalence declaration specifies that two or more attributes from different view objects should be equivalent, i.e., treated as one shared attribute.

Consider the following simple example where a temperature sensor is represented by a multi-view object with three views. The first view represents the sensor in the process schematic, i.e., in the topological view of, say, the main system. The second view represents how the sensor is physically realized, in this case as a thermo-element that is connected to an analog input at the controller IO card. The third view represent the sensor in a causal signed digraph model that is used for on-line fault diagnosis. The class structure for the temperature sensor example is shown in Fig. 10. Class definitions for ordinary objects are shown as empty triangles. Class definitions for multi-view classes are shown as triangles with three circles in them. In this example we assume single inheritance only, i.e, we have two class hierarchies. T44 is an instance of the multi-view class *Diagnosed-temperature-sensor.*

Multi-view objects are also used to handle the problem with multiple hierarchical decompositions of a plant. Since each view of a multi-view object can be a composite object it is possible for a multi-view object to have different internal structure in its different views. Note that the multi-view objects themselves are not part of the object hierarchy, except indirectly through their views. The knowledge base is structured into a set (forest) of hierarchies (trees). Each tree contains a hierarchical decomposition of the plant that fits a certain application. Hence, the different trees can have different structure. However, generally their overall structures resemble each other, e.g., the root object in each tree is usually a description of the entire plant or some system in the plant whereas the leaf nodes are descriptions of the process components. The multi-view objects bind together objects from the different trees as a spider's web according to Fig. 11.

The motivation for multi-view objects is the need to have multiple representations in the knowledge base for the same object, in the same time as the entire object with all its representations is represented by one object. The solution usually applied in the object-oriented community for an object that in the same time can be seen as different objects is multiple inheritance. Using multiple inheritance each part of the object can be defined in a separate class definition. The entire object is then defined by a class that inherits from all the separate classes. The difference with between this and multi-view objects is that an instance of a class derived from multiple classes is still only one object that only may appear in one context. In Struss (1987) an alternative implementation of views is described. Here views are used to represent different aspects of process components, e.g., electrical aspects, thermal aspects, mechanical aspects, etc. All object attributes, connections, and connection terminals are parametrized by views. Different views can also have different internal structure.

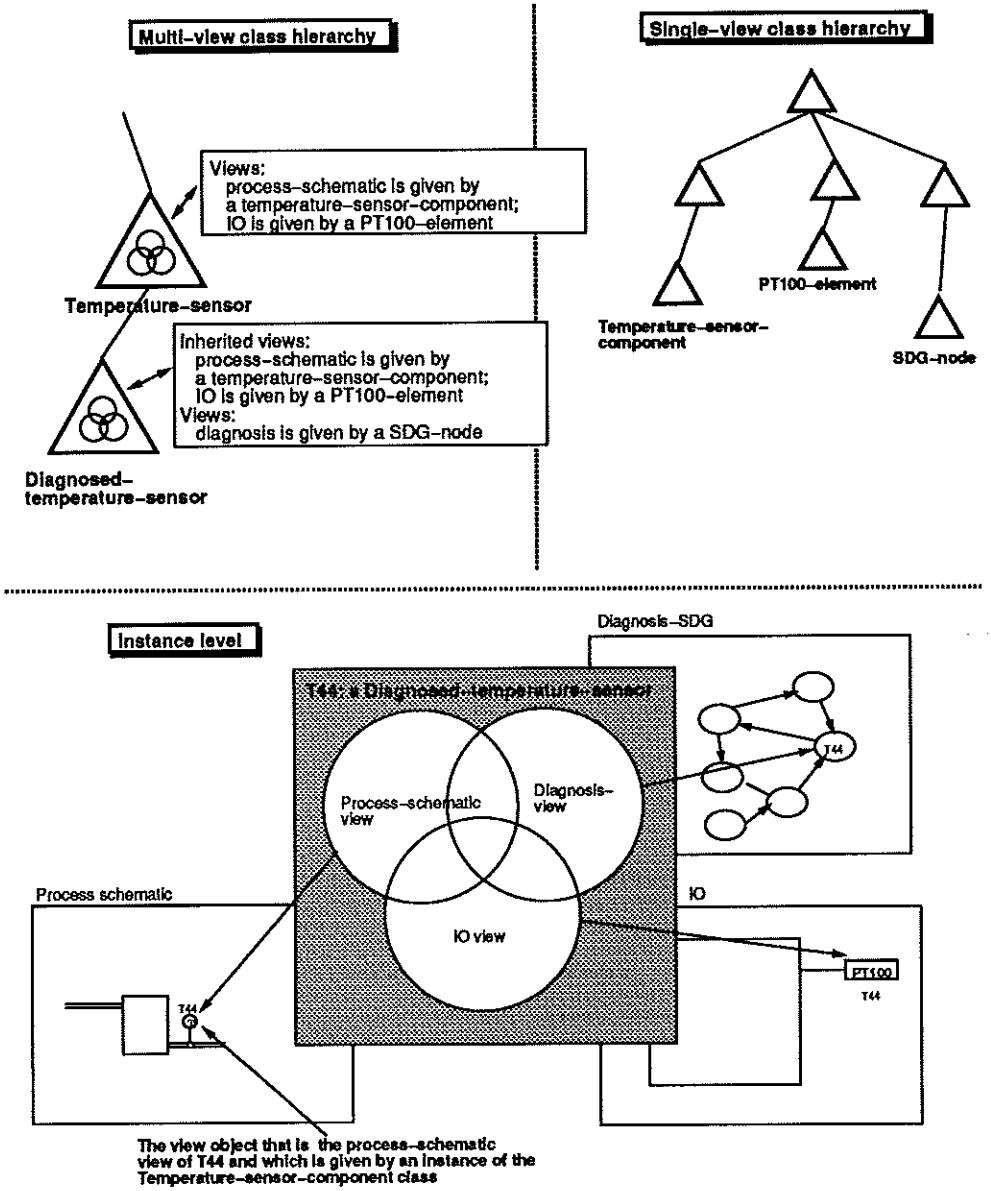

Figure 10. Multi-view classes and objects for a temperature sensor

4. Demonstrator

In order to better visualize the KBCS concept a demonstrator has been developed. The demonstrator has been implemented in G2. Although G2 was originally developed as a real-time expert system environment the current functionality of G2 allows it to be used as a general prototyping tool for general on-line applications combining conventional programming techniques with knowledge-based programming techniques. The goals of the demonstrator is to

- visualize the KBCS concept by
  - mimicing the internal structure of a KBCS knowledge base in G2 including multi-view objects, systems, etc.,
  - showing the different interfaces to the KBCS, i.e., the knowledge base browser and the different user interfaces;

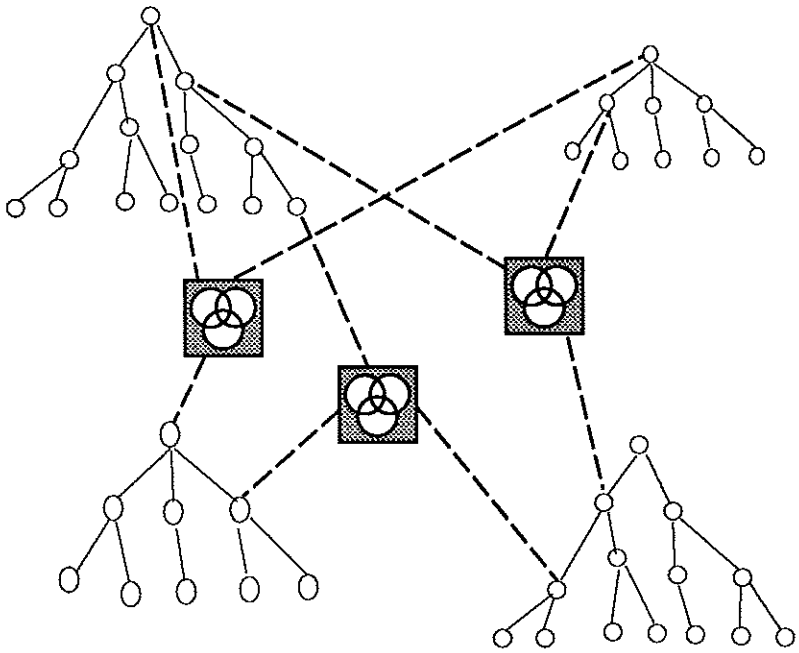

Figure 11. Multiple hierarchies

- show how the concept can be used to structure information and knowledge concerning a realistic, industrial process, and
- show how different applications can be implemented in the KBCS.

The demonstrator does not show the distributed aspects of the KBCS concept, i.e., how executable code is extracted from the knowledge base, compiled, and distributed to controller units. Also the demonstrator does not show the designer's interface to the knowledge base and the tools that support the designers entering information into the knowledge base.

Steritherm

Steritherm from Alfa-Laval is used as the example process in the project. Steritherm is a process for UHT (Ultra High Temperature) sterilization. A UHT product is a liquid that has been subjected to a continuous flow heating process at a high temperature for a short time, normally 135-140 degrees C for a few seconds. The purpose of the process is to kill all micro organisms in the product. If the sterilized product is packed during aseptic conditions, it can be stored at room temperature during a long time. The most common product is milk but it is also possible to process cream, coffee, dressings, sauces, etc. In Steritherm the product is heated indirectly in plate heat exchangers until the temperature reaches the sterilization temperature. The product is maintained at this temperature for a few seconds. Then it is cooled in further heat exchangers and pumped to the packing machine. Fig. 12 shows a schematic of Steritherm. From a system point of view Steritherm contains a main system, a warm water system, a cold water system, a steam system, two electrical systems, a pneumatic system, and a control system.

Steritherm is a continuous process with large sequential elements. It has four main phases: sterilization, production, intermediate cleaning, and final cleaning. Before production starts the process is sterilized. This is done by pumping hot water through the main system. Normally production

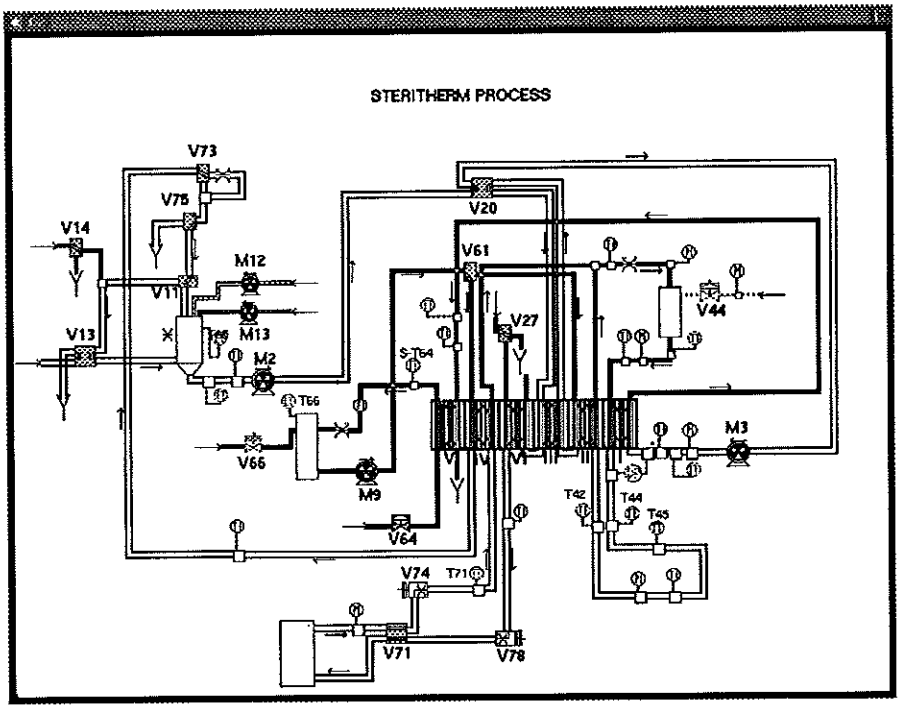

Figure 12. Steritherm schematic (taken from the G2 demonstrator).

goes on for 6 to 8 hours before the burn-on that takes place in the heat exchangers has reached the level where production has to be stopped and the process cleaned. Intermediate cleaning is a fast but not so accurate cleaning procedure that does not require the process to be resterilized.

The G2 demonstrator

The G2 demonstrator consists of the following parts:

- the knowledge base,
- the operator interface,
- the process engineer interface, and
- the real-time simulation model of Steritherm.

The operator and process engineer interfaces consist of a set of pre-defined pictures that have been designed to meet the demands of the different user categories. In addition to their custom interfaces both the operator and the process engineer can chose to look at the contents of the entire knowledge base through the knowledge base browser. The operator interface is partly a subset of the process engineer interface, i.e., the process engineer has access to all the information that the operator has.

Demonstrator Scenario: Although Steritherm is the focus of the demonstrator, the actual scenario consists of a small dairy. The dairy contains three Steritherm process (of which only one is simulated) with different capacities. The products are packed by five packing machines, see Fig. 13. The dairy can produce four different UHT products: milk, low fat milk, strawberry flavored milk, and low fat strawberry flavored milk.

The dairy is controlled by a distributed KBCS consisting of two supervisory units (KBSU), four controller units (KBCU), two operator stations, one

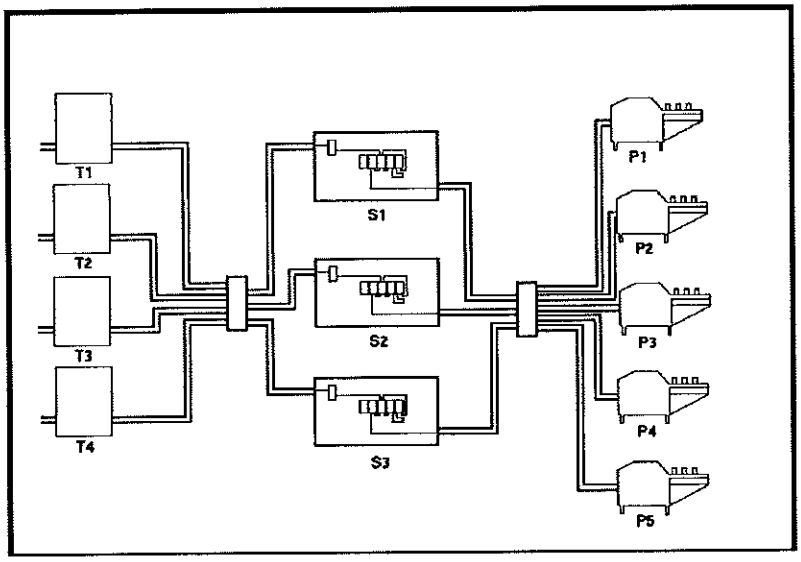

Figure 13. Dairy scenario

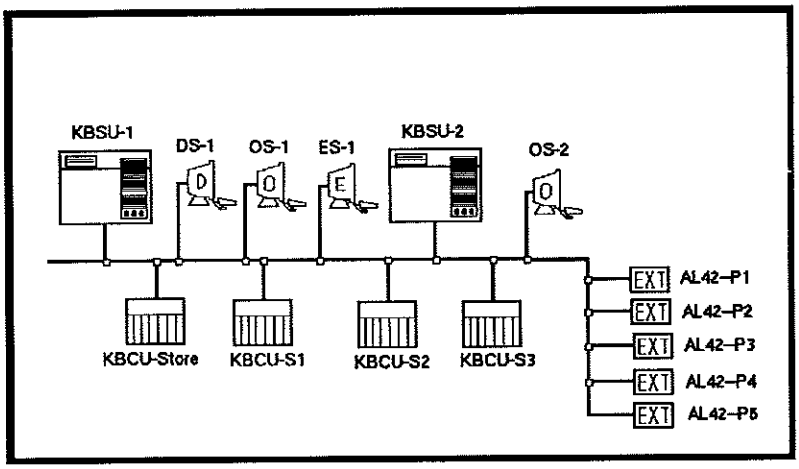

Figure 14. KBCS scenario

process engineer station, and one designer station. The supervisory units perform the supervisory control, monitoring, and planning functions of the KBCS. They also contain the knowledge base. The four controller units are dedicated to Steritherm-1 (S1), S2, S3, and the raw product storage tanks. It is assumed that the packing machines have their own controllers that are only loosely interfaced to the KBCS. The situation is shown in Fig 14.

Steritherm simulator: A real-time simulation of one of the Steritherm processes implemented using G2's built-in simulation facilities is used as a replacement for a real process (Christiansson and Ericsson, 1989). The simulation is object-oriented with objects representing heat exchanger sections, pumps, valves, tanks, sensors, etc. The flow and pressure simulation is static and the temperature and level simulation is dynamic. The mechanical regulators that are part of a Steritherm process, e.g., constant pressure valves, are also simulated. In the simulation it is possible to introduce various sensor and process faults. Altogether around 400 variables are simulated. The static properties of the simulation model have been tuned against real process data.

The Knowledge Base

The knowledge base part of the G2 demonstrator consists of:

- the Class definitions,
- the Multi-view class definitions, and
- the Object Space.

The class definitions contain the definitions for all the "ordinary objects" in the knowledge base. Ordinary objects are implemented as G2 objects. Hence, the class definition for knowledge base objects is equivalent to the G2 class definition. The class definitions are organized into a graphical tree structure that shows the inheritance hierarchy. The knowledge base contains 250 class definitions.

Multi-view class definitions contain the definitions for the multi-view objects. Multi-view class definitions are organized into a tree structure in the same way as ordinary class definitions. Multi-view objects are not directly supported by G2. A multi-view object and the objects representing the different views of the multi-view object are all G2 objects. The relationships between a multi-view objects and its views are represented by G2 *relations*, a built in feature for expressing relationships between objects. The number of multi-view class definitions and, hence, the number of multi-view objects is relatively small. The reason for this is that only a limited number of multi-view objects have been modeled.

The object space contains the instances, i.e. it contains the multi-view objects representing process units or components, and the "ordinary" composite objects that are the roots in the hierarchical decompositions of the plant. Multi-view objects representing systems or subsystems are contained in the multi-view objects representing the process units.

Composite objects are represented as G2 objects that have subworkspaces. The internal structure of the composite object is represented as interconnected objects placed at the subworkspace.

When the knowledge base is inspected through the browser the user has access to set of navigation tools. It is possible to zoom in on a composite object and have the subworkspace of the object shown, thus moving one hierarchical level down. It is also possible to move upwards in the decomposition hierarchy either one level at a time or multiple levels at a time. It is possible to move between different views of a multi-view object and to the multi-view object itself. From both "ordinary" objects and multi-view objects one can go to the class definitions. By selecting a class definition one gets a menu of all instances of the class. Finally, it is possible to move chronologically backwards to the 10 last visited objects in the knowledge base.

The top-level object in the knowledge base is a multi-view object representing the entire dairy. The dairy contains multi-view objects representing the main system, the 10kV electrical system, the 380V electrical system, the control hardware system, and the control software system. The number of systems and the number of views for the systems at the dairy level are quite small. Steritherm-1 is also represented by a multi-view object. Here the system and view structure is richer as shown in Fig. 15.

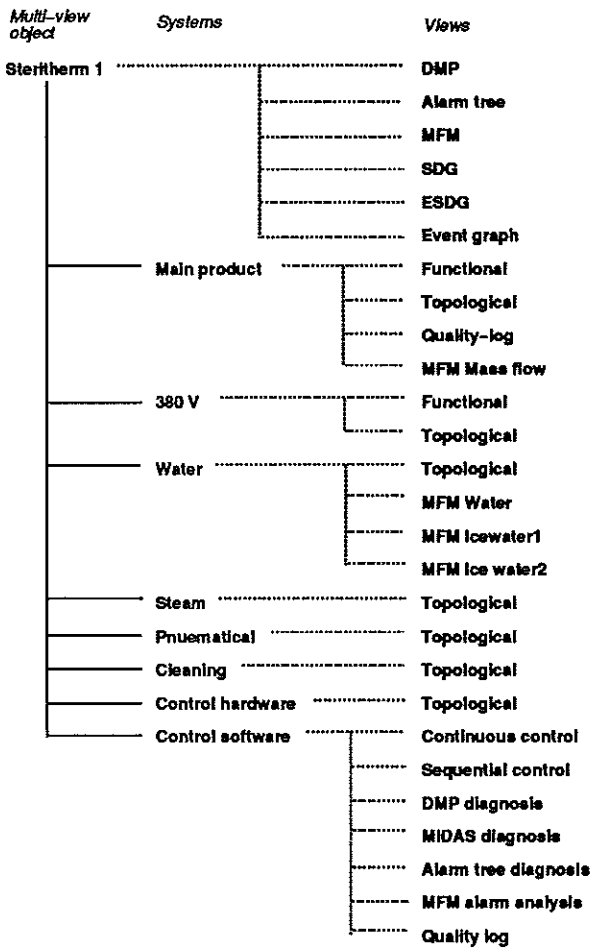

Figure 15. The Steritherm-1 multi-view object

The other objects that have been modeled as multi-view objects are a few process components only, e.g., some heat exchanger sections, a temperature transmitter, a pump, a valve, etc. This is by far not a complete list of what really ought to be modeled as multi-view objects. However, they serve as a set of examples of how multi-view objects could be used.

The Operator Interface

The operator interface is one example of what a user interface might look like. The basic idea is that the information presented in the operator interface is generated from the contents of the knowledge base.

The organization of the operator interface is shown in Fig. 16. The top panel contains a set of mailboxes where warning messages, alarms, production messages, system messages, etc., are indicated. By clicking on the mail box the operator opens it and there he can read the messages, scroll through the contents of the mailbox, acknowledge messages, etc. The right hand side panel contains buttons. By clicking on the buttons the user selects the picture he wants to see in the picture area.

Currently the operator can select to see six different pictures. In Fig. 16, a process schematic of Steritherm-1 is shown in the picture area. The other pre-defined pictures are a dairy overview schematic, a matrix display showing the interconnections between the dairy units, a matrix display showing

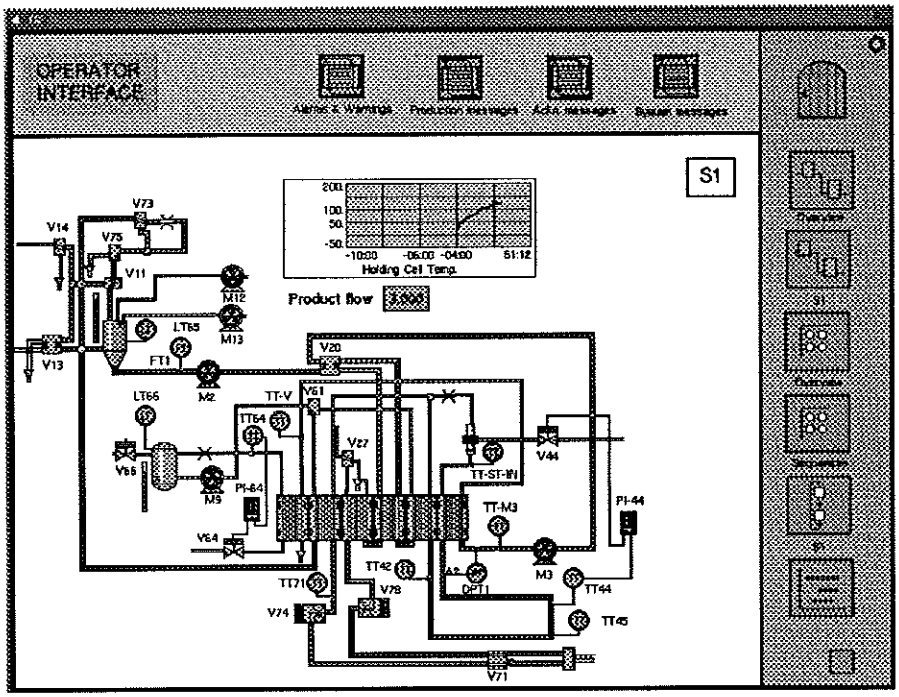

Figure 16. Operator interface set-up

sequence control information about the Steritherms, the sequential function chart containing the sequence control logic for Steritherm-1, and a production scheduling display showing the output of the production scheduling presented on Gantt diagrams.

By clicking on the top-most button (the door) the operator enters the knowledge base browser. It is also possible for the operator to enter the knowledge base at a specific place. For example, from the PID controllers in the Steritherm schematic it is possible to enter the knowledge base at the PID object.

The Process Engineer Interface

The process engineer interface has the same organization as the operator interface. The difference is that the process engineer has access to more information. In addition to the operator pictures the process engineer can see a signal analysis display where he can plot different signals, a product following display that shows the output from the product following application, a diagnosis display that shows more detailed information about the on-line monitoring and diagnosis applications, and a hardware display through which he can check the status of hardware units, communication units, etc.

The process engineer interface is shown in Fig. 17. The picture area here shows the Steritherm-1 sequential function chart.

Applications

A number of control, diagnosis, and scheduling applications have been implemented. The focus has been on non-conventional approaches to on-line model-based diagnosis. Three such applications have been implemented: DMP, MIDAS, and MFM.

The applications are:

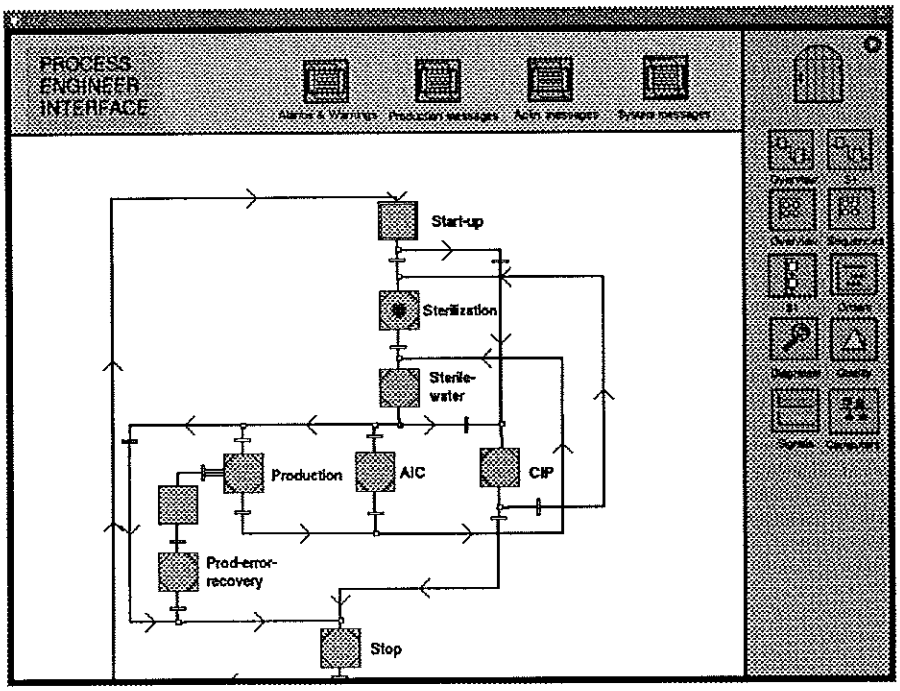

Figure 17. Process engineer interface

- PID control,
- sequence control implemented using Grafcet-style sequential function chart formalism,
- rule-based burn-on monitoring,
- model-based diagnosis based on quantitative governing equations according the the Diagnostic Modeling Processor (DMP) method,
- model-based diagnosis based on qualitative causal digraphs and event models according to the MIDAS formalism,
- Multi-level Flow Model (MFM) based alarm analysis,
- fault tree based off-line diagnosis,
- heuristic production scheduling, and
- a product following system.

PID control: The PID controllers are implemented as G2 objects. The PID algorithm is represented by a G2 procedure that is executed when the controller is sampled.

Grafcet: The sequence control logic has been implemented using Grafcet formalism (David and Alla, 1992). Steps are represented as objects. The steps contains actions that are performed when a step is active. Examples of actions are open a valve, start a pump, etc. Steps can have an internal structure of substeps, i.e., they can be macro-steps. Transitions are also represented as objects. Associated with each transition is a condition that is tested when the step preceding the transition is active. When the transition condition becomes true the preceding step is deactivated and the step succeeding the transition becomes active. Alternative and parallel paths are allowed (Årzén, 1991b).

Rule-based monitoring: G2 rules are used to monitor the burn-on that occurs in the heat exchangers. Burn-on is detected by monitoring the mean values and trends of differential pressure sensors and control signals. This is expressed as G2 rules that operate upon time histories.

DMP: The Diagnostic Model Processor method (DMP) (Petti and Dhurjati, 1991; Petti 1992) is based on quantitative constraint equations called model equations written on residual form. The equations may contain anything that can be calculated at run-time including past sensor values, mean values, etc. Usually the equations represent some kind of static or dynamic balance equation.

Associated with each equation are tolerance limits which represent the expected (fault free) upper and lower values of the residual for which the equation is considered satisfied. Also associated with each equation is a set of assumptions which if satisfied guarantee the satisfaction of the equation, i.e., the residual is close to zero. The model equations are updated continuously. Since the residuals are not uniform in magnitude, they are transformed into a metric between -1 and 1 which indicates the degree to which the model equation is satisfied: 0 for perfectly satisfied, 1 for severely violated high, and -1 for severely violated low.

Model equations and assumptions are represented as objects. The dependence of a model equation on an assumption is indicated by a dependence connection. Associated with the dependence is sensitivity information, i.e., sensitivity derivatives or heuristically chosen numbers, that indicate how sensitive an equation is to a deviation in an assumption.

Conclusions about the satisfaction of each assumption (fault state) is made by combining the evidence from the model equations. This so called failure likelihood is calculated for each assumption as a weighted sum of the transformed residuals of the model equations that the assumption affects with the sensitivities taken as the weights. The failure likelihood is interpreted as indicating a likely condition of the assumption failing high as the value approaches 1 and a likely condition of the assumption failing low as the value approaches -1.

DMP allows the detection of non-competing multiple faults. It is based on taking snapshots of the plant status and checking the satisfaction of dynamic and static quantitative balances. It has relations both to parity space approaches to fault detection (Gertler, 1991) and artificial neural networks.

The DMP models of Steritherm contains 25 model equations and 17 assumptions. The DMP network, shown in Fig. 18 , is represented as one view of the Steritherm-1 multi-view object.

MIDAS: MIDAS (Model Integrated Diagnosis Analysis System) is an approach to model-based, on-line diagnosis developed at MIT by Kramer, Oyeleye, and Finch, (Oyeleye, 1989; Finch, 1989). MIDAS is based based on qualitative causal reasoning about deviations from a nominal steady-state. MIDAS starts with a library of signed directed graph (SDG) models for different process component. An SDG describes the variables of the process components as nodes and the qualitative relationships between the variables as arcs between the nodes. Nodes have the qualitative states $0, +$, or $-$ depending on if the process variable is zero, positive, or negative. An arc between two nodes n_1 and n_2 has the sign $+(-)$ if an increase in n_1 causes

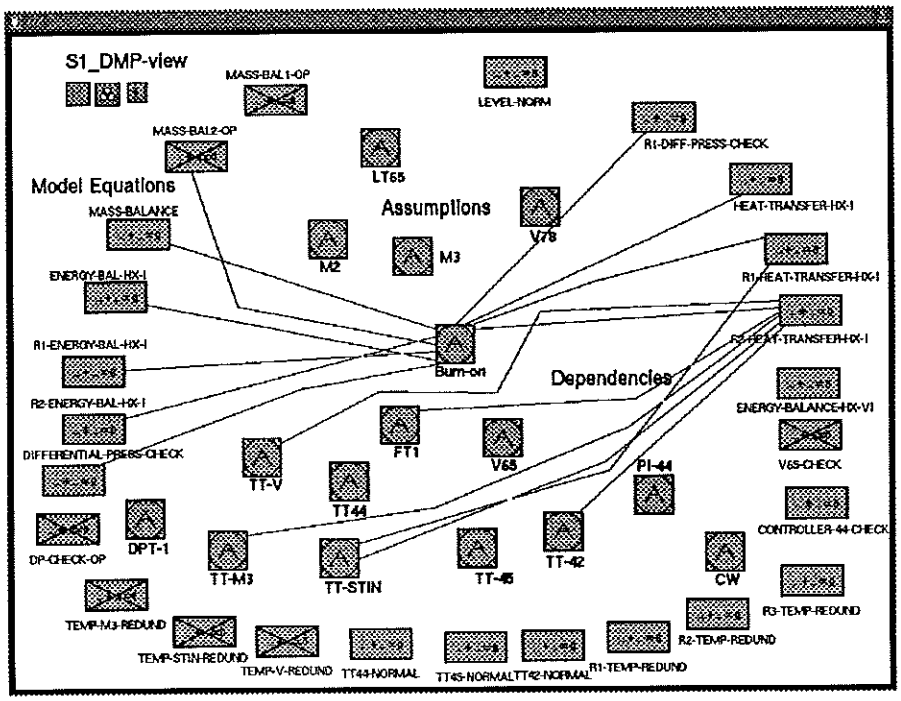

Figure 18. The DMP model with only a few of the dependencies shown. The model equations are arranged in a circle. The assumptions are the objects in the center with an A in them.

an increase (decrease) in n_2. Included in the SDG is also the set of root causes that might affect the process.

Using the SDG model library an SDG model for the entire process is constructed. A problem with SDG models is that they are ambiguous. In many situations it is impossible to decide whether an variable really will increase or decrease. To overcome this problem the SDG model is analyzed and additional non-physical arcs are inserted into the graph, thus converting it to an Extended SDG (ESDG). The actual model that MIDAS uses on-line during diagnosis is an event graph model. This can be derived automatically from the ESDG. The event graph consists of events, i.e., qualitative state transitions, root causes, and links connecting events with other events and with the root causes. The qualitative state of a variable is either high, normal or low, compared to the nominal steady-state value. That is, for every measured variable, i.e., sensor, four events are possible: a transition from normal to high, from high to normal, from normal to low, and from low to normal. Each root cause is connected to the event that should be the first symptom of the fault. The links between the events express possible fault propagations, i.e., chains of events. The links may have conditions attached to them telling when they are valid and what diagnostic conclusions that may be drawn when two events are linked together. The Steritherm event graph is shown in Fig. 19.

Every sensor has an associated monitor which is responsible for detecting state changes, i.e. events, concerning the variable. The monitors contain information about the threshold levels, hysteresis, filtering, etc. When an event has been detected it is sent to the event interpreter. It is also possible for the event interpreter to perform interrogation, i.e., ask a monitor to predict an event that has not yet occurred. When the event interpreter

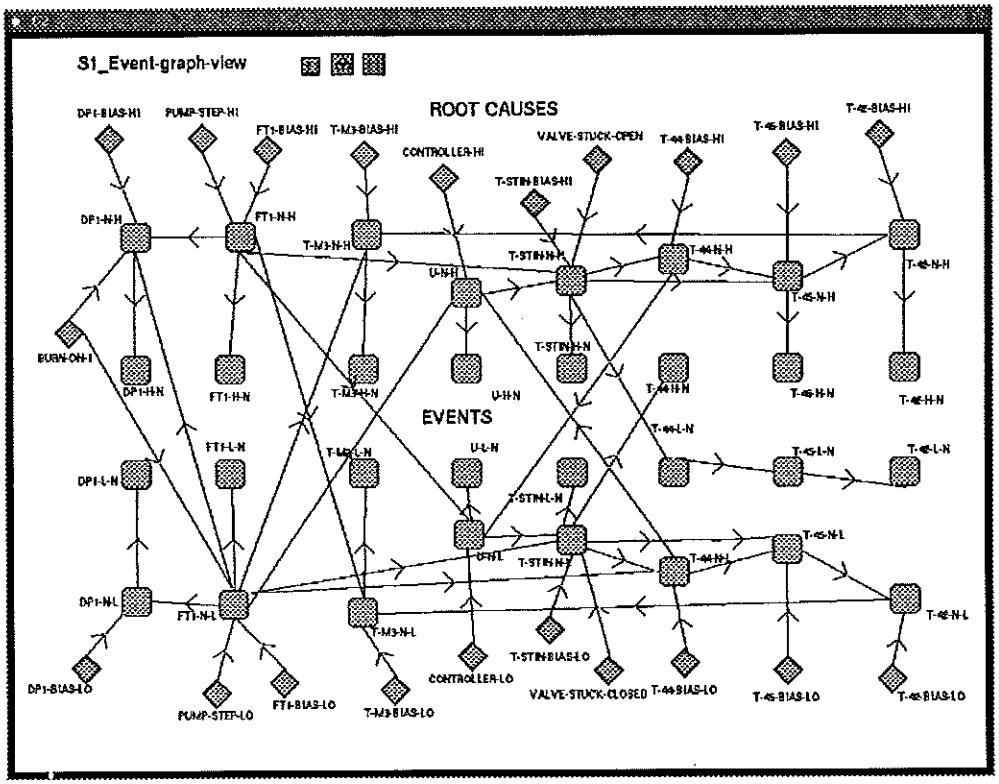

Figure 19. Steritherm event graph model.

receives an event it tries to, based on the information in the event graph, link the new event with clusters of old events. The clusters also contain hypothesized root causes that may have caused the events in the cluster. Based on incoming events clusters can be merged or split. The set of clusters is called the hypothesis model. The off-line modeling part of MIDAS and the on-line diagnosis is shown in Fig. 20.

In the MIDAS model only about two thirds of Steritherm has been modeled. The SDG consists of 68 nodes and 11 root causes. The SDG, ESDG, and event graph models are all different views of the Steritherm-1 multi-view object. A detailed description of the G2 implementation of MIDAS is given in (Nilsson, 1991). A comparison between DMP and MIDAS is given in (Nilsson *et al*, 1992).

MIDAS is based on qualitative causal reasoning. It can handle multiple non-competing faults. On purpose the order of events or time between events is not taken into account during the diagnosis.

MFM-based alarm analysis: Multi-level flow models (MFM) (Lind, 1990) is a modeling technique used to explicitly represent means-end information. A process is described in terms of the goals that the process is intended to fulfill, the abstract functions available for fulfilling the goals, and the components realizing the functions. Goals are associated with functions or network of functions through achieve relations and functions can be conditioned by other goals via condition relations. MFM contains a number of pre-defined energy and mass flow functions including sources, transports, barriers, storages, balances, sinks, etc. These are described by a graphical language.

MFM models can be used as the basis for various monitoring and diagnosis

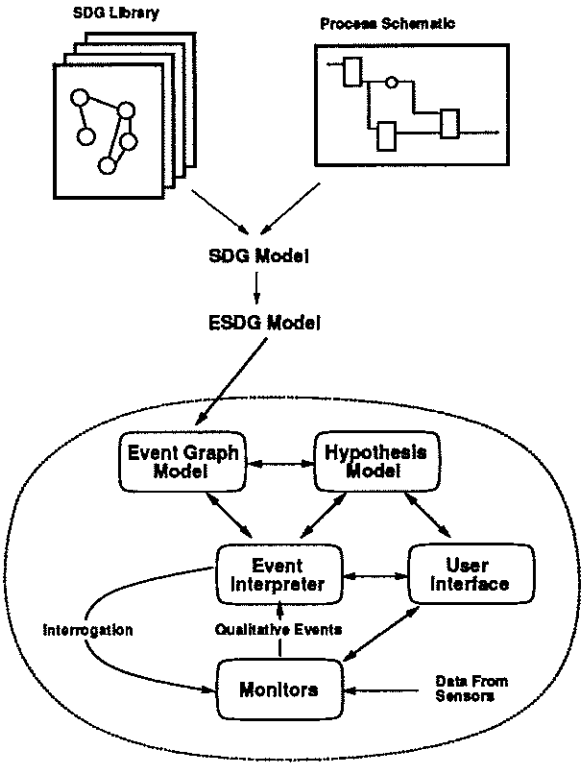

Figure 20. MIDAS model generation and on-line structure (inside the dashed line)

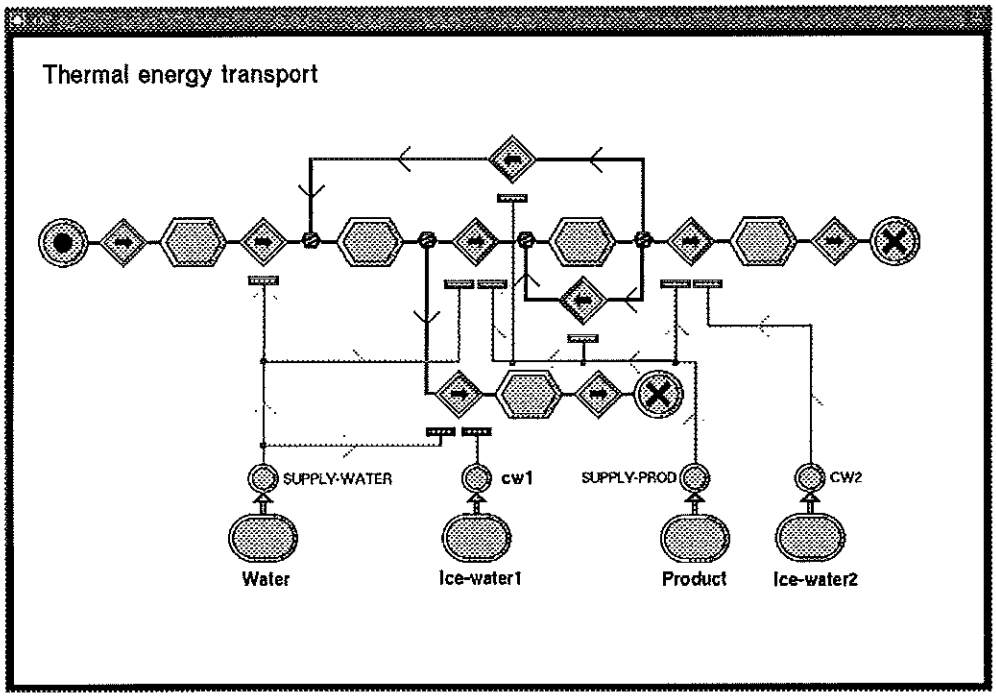

Figure 21. The thermal energy network of Steritherm.

tasks, e.g., measurement validation (Larsson, 1992a), fault diagnosis (Larsson, 1992b), and alarm analysis (Larsson, 1991). The demonstrator contains an MFM model of Steritherm that is used for alarm analysis. The flow network representing the thermal energy flow part of this model is shown in Fig. 21.

The task of the alarm analysis is to decide if an alarm is a primary alarm or if might be a secondary alarm. It is based on a set of generic assumption on how faults can propagate from flow function to flow function. Some primary faults in some types of flow functions may cause secondary faults in the connected functions, while faults in others may not. When the alarm state of a function is unknown the missing value is deduced using a consequence propagation strategy.

The MFM model is represented as one view of Steritherm-1. It contains as subparts the MFM mass flow view of the main system and the MFM mass flow views of the water system.

Fault tree based diagnosis: When a serious alarm occurs in a Steritherm process the safety logic in the control system automatically shuts down the process. The task of finding the cause of the alarm is therefore performed off-line. The fault tree based diagnosis application uses fault trees that associate an alarm with possible function faults such as a fault in the steam system, a pressure fault, etc., and that associate the function faults with the physical component faults that may have caused the function faults. The fault trees are represented graphically by interconnected fault objects. They are traversed giving advice and instructions to operators or maintenance personnel doing the troubleshooting.

Heuristic production scheduling: The production scheduler is a simple scheduler application where the aim is to correctly schedule orders coming to the dairy. An order is represented by a G2 object with attributes containing information about the order number, the product type, the customer name, the volume of the order, the package size of the product, the paper quality, the deadline when the order must be delivered, the accumulated volume that has been produced of an order, and the priority of the order.

The scheduler's output is a schedule that specifies which sterilization and packaging machines should be used, and at what times, to produce each order. The schedule tries to meet deadlines and to use the sterilization machinery as efficiently as possible. The schedule is presented in two Gantt diagrams, one showing the coarse schedule for the coming week and the other showing the detailed schedule for the current day. Colour is used to show what type of milk that is produced and the different operating modes of the Steritherms (sterilization, production, AIC, CIP). The daily schedule is shown in Fig. 22. The x-axis shows the hours of the day and the y-axis the production resources, i.e., the Steritherms and the packaging machines.

Product following: A simple product following system has been developed. Its intended use is quality follow-up. The idea is to be able to record how the milk has been treated as it flows through the process. Objects representing a certain volume of milk are dynamically created at the inlet to Steritherm, with the creation rate depending on the production. The objects have attributes representing temperatures, pressures, etc., in the different parts of the process. Originally the attributes have no values. As the milk flows through the process the attributes are gradually filled in with actual processing data.

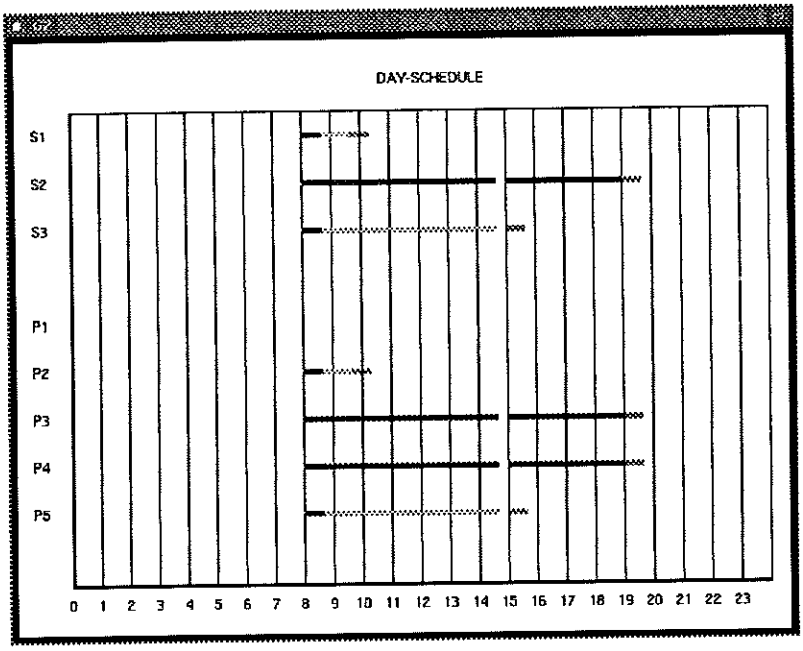

Figure 22. Daily schedule

Demonstrator summary

The goal of the demonstrator is to give a vision of how a future KBCS might look like. Several control, monitoring, and diagnosis applications have been implemented. The main motivation for the applications have been to test how the different process models that the methods are based on fit with the object–system–view knowledge base structure. In many cases the functionality of the applications overlap, i.e., both DMP and MIDAS perform on-line diagnosis and are both capable of diagnosing essentially the same type of faults. In a real system probably only one diagnosis method would be implemented. The demonstrator has also served as testbench in which alternative diagnosis have been evaluated. This has a strong value in itself.

5. Conclusions

A new concept for DCSs has been outlined. The concept is based on a database or knowledge base containing different descriptions or models of the plant including the control system. The paper has focused on the structuring mechanisms needed in the knowledge base. The plant description is centered around the systems in the plant by which is meant the flows of mass, energy, and information. In order to handle information that must be represented in more than one way at the same time the concept of multi-view objects has been defined.

To verify and visualize the concept a demonstrator has been implemented. The demonstrator contains the knowledge base and two user interfaces. The demonstrator has been applied to the control, monitoring, diagnosis, and scheduling of a small dairy consisting of Steritherm processes. The demonstrator applications has focused on different approaches to model-based on-line diagnosis. It has been possible to evaluate different approaches to diagnosis on the same process.

The development of a new DCS generation is a major involvement. Requirements on backward compatibility makes it unlikely that a new DCS will be developed from scratch. Instead DCS system will hopefully gradually develop along the direction pointed out in the project. The development of integrated DCSs is not an issue for the traditional DCS developers only. It clearly also affects the developers of CAD systems, documentation systems, etc. Therefore it is important that standards are defined that allows an open system approach. A step in this direction is the Express language for data modeling (Schenck, 1990) and the CALS project.

A question of more philosophical nature is if it at all possible to gather "all" plant information in a single database. It is quite clear, even from the relatively small Steritherm example, that the amount of information is enormous. However, an integrated DCS would be of great value also if only parts of the information would be included, e.g., the information concerning only the plant operation.

Acknowledgements

The author would like to thank all the participants in the KBRTCS project and in particular Claes Rytoft, Anders Åberg, Nick Hoggard, and Martin Uneram of ABB Corporate Research; Oddbjørn Evjen of ABB Automation; Jan Eric Larsson, Anders Nilsson, and Karl Johan Åström of the Department of Automatic Control, Lund; Tom Petti of the University of Delaware; and, finally, Marcel Schoppers of ADS.

References

ÅRZÉN, K.-E. (1989): "Knowledge-based control systems: Aspects on the unification of conventional control systems and knowledge-based systems," Proceedings of the 1989 American Control Conference (ACC '89), Pittsburgh, Pennsylvania.

ÅRZÉN, K-E. (1990): "Knowledge-based control systems," Proceedings of the 1990 American Control Conference (ACC '90), San Diego, CA.

ÅRZÉN, K-E., C. RYTOFT, and C. GERDING (1990): "A Knowledge-Based Control System Concept," Proceedings of the 1990 European Simulation Symposium, Ghent November 8–10, SCS International.

ÅRZÉN, K-E. (1991a): "Knowledge based applications in the process industry: Current state and future directions," Keynote address, Proceedings of the IFAC Workshop on Computer Software Structures Integrating AI/KBS in Process Control, Bergen May 29–30.

ÅRZÉN, K-E. (1991b): "Sequential function charts for knowledge-based, real-time applications," Proc. of Third IFAC Workshop on AI in Real-Time Control, Rohnert Park, CA.

CHRISTIANSSON, M., and P. ERICSSON (1989): "Knowledge-based control and modelling with G2," Master thesis TFRT-5411, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

DAVID, R. and H. ALLA (1992): Petri Nets and Grafcet: Tools for Modeeling Discrete Event Systems, Prentice Hall.

ELMQUIST, H. (1991): "A uniform architecture for distributed automation," Proc. of the ISA/91, Anaheim, CA.

FINCH, F.E. (1989): "Automated Fault Diagnosis of Chemical Process Plants using Model-Based Reasoning," Sc.D. Thesis, Massachusetts Institute of Technology.

GERTLER, J. (1991): "Analytical redundancy methods in fault detection and isolation – Survey and synthesis," Preprints of the IFAC SAFEPROCESS Symp., Baden-Baden, Germany.

LARSSON, J.E. (1991): "Model-Based Alarm Analysis Using MFM," Proceedings of the 3rd IFAC International Workshop on Artificial Intelligence in Real-Time Control, Rohnert Park, CA.

LARSSON, J.E. (1992a): "Model-Based Measurement Validation Using MFM," Preprints of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries, University of Delaware, Newark, Delaware.

LARSSON, J.E. (1992b): "Model-Based Fault Diagnosis Using MFM," Preprints of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries, University of Delaware, Newark, Delaware.

LUKAS, M.P. (1986): Distributed Control Systems: Their evaluation and design, Van Nostrand Reinhold Company Inc., New York.

MOORE, R. L., H. ROSENOF, and G. STANLEY (1990): "Process Control Using a Real-Time Expert System," Proceedings of the 11th IFAC World Congress, Vol 7, Tallinn, Estonia, pp. 234–239.

NILSSON, A. (1991): "Qualitative Model-Based Diagnosis – MIDAS in G2 TFRT-5443, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden,".

NILSSON, A., K-E. ÅRZÉN and T.F. PETTI (1992): "Model-Based Diagnosis – State Transition Events and Constraint Equations," Submitted to the IFAC Symposium on AI in Real-time Control, Delft.

OYELEYE, O.O. (1989): "Qualitative Modeling of Continuous Chemical Processes and Applications to Fault Diagnosis," Sc.D. Thesis, Massachusetts Institute of Technology.

PEKNY, J., V. VENKATASUBRAMANIAN and G.V. REKLAITIS (1991): "Prospects for computer-aided process operations in the process industries," in L. Puigjaner and A. Espuna (Eds.): Computer-Oriented Process Engineering, Elsevier Science Publishers B.V., Amsterdam.

PETTI, T.F. and P.S. DHURJATI (1991): "Object-based automated fault diagnosis," Chem. Eng. Comm., **102**, 107–126.

PETTI, T.F. (1992): "Using Mathematical Models in Knowledge Based Control Systems," Ph.D. Thesis, University of Delaware.

REKLAITIS, G.V. and H.D. SPRIGGS (Eds.) (1987): Foundations of Computer Aided Process Operations, CACHE/Elsevier.

SCHENCK, D. (1990): "Express language reference manual," ISO TC184/SC4/WG1 N466.

SMITH, J. M. (1990): CALS: The strategy and the standards, The Camelot Press, Trowbridge, Wiltshire, England.

STRUSS, P. (1987): "Multiple representation of function and structure," in J. Gero (Ed.): Expert Systems in Computer-Aided Design, Elsevier Science Publishers B.V. (North-Holland).