CODEN:LUTFD2/(TFRT-5411)/1-80/(1989)

# Knowledge-based control and modelling with G2

---

## An IT4 project

Michael Christiansson
Pär Ericsson

Department of Automatic Control
Lund Institute of Technology
October 1989

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

*Document name*
Master Thesis

*Date of issue*
October 1989

*Document Number*
CODEN: LUTFD2/(TFRT-5411)/1-80/(1989)

*Author(s)*
Michael Christiansson
Pär Ericsson

*Supervisor*
Christer Gerding SattControl, Karl-Erik Årzén LTH

*Sponsoring organisation*
IT4

*Title and subtitle*
Knowledge-based control and modelling with G2.

*Abstract*

This report describes a knowledge-based real-time control system. The project is a part of the IT4 project "knowledge-based real-time control systems". The UHT-process Steritherm from Alfa-Laval is used as a demonstrator. The system is developed with an expert system tool for real-time applications called G2 from Gensym Corp.

Since an actual Steritherm process is not physically available a simulation model of the process has been developed within G2. The model simulates numerical values of pressures, flows, temperatures and levels.

The purpose of this project is to examine the inner structure of the knowledge-base and how well G2 can implement this. The control system contains the following parts: a schematic that represents the process seen from the control system, regulators, a sequence net for handling the different sequences of the process, alarm-analysis for helping service personnel locating faults in the process, and reule-based supervision of the burn-on in the process. The knowledge-base is hierarchically structured in objects describing the process at different degrees of resolution. Views are used at each hierarchical level, to represent different types of information about the process.

*Key words*

*Classification system and/or index terms (if any)*

*Supplementary bibliographical information*

*ISSN and key title*

*ISBN*

*Language*
English

*Number of pages*
80

*Recipient's notes*

*Security classification*

# Preface

Knowledge-based programming techniques give possibilities to develop control systems with completely new functions. To do this, methods and principles used in conventional control systems have to be integrated with new technologies for knowledge-based processing.

This report is the result of a project aiming to construct a control system with this new technique. The project is a part of the IT4 project "Knowledge-based real-time control systems". The UHT-process Steritherm from Alfa-Laval is used as a demonstrator.

The system has been developed with G2, a real-time expert system from Gensym Corp on a Symbolics 3650 Lisp machine.

The work has been performed during June - October, 1989 at the Department of Automatic Control, Lund Institute of Technology in cooperation with SattControl and ABB. During the work we have been employed by SattControl.

**Acknowledgement**

We want to thank Karl-Erik Årzen, Department of Automatic Control, Lund Institute of Technology, for his deep support and help with the construction of the system. He has read and corrected many versions of this report. Without his help, this project would probably not have come to an end.

We also want to thank Christer Gerding, SattControl, who has supported us with material concerning the Steritherm process, read different versions of the report, and corrected a lot of errors and Claes Rytoft, ABB, who has supported us with material and information about how the Steritherm process really works. Finally, we want to thank Thomas Petti, Univeristy of Delaware, for reading the final version of this report and correct the last errors.

*Michael Christiansson* *Pär Ericsson*

Lund October 1989

# Contents

# 1 Introduction

## Background

This report is the textual documantation of the masters thesis project "Knowledge-based control and modelling with G2". The project is performed as a part of the IT4 project "Knowledge-based real-time control systems" (IT4, 1988), jointly performed by ABB, SattControl, and the Department of Automatic Control, Lund Institute of Tecnology.

Within the IT4 project, a concept for integration of knowledge-based techniques with conventional process control systems is examined. The concept is built upon a common knowledge-base containing knowledge and information about the process and the control system. The knowledge-base is hierarchically structured in objects describing the system at levels ranging from an overall "plant" level down to process component objects. For each object different types of knowledge must be represented. This is done using multiple views or representation of the objects. Some examples of different representations are a structural representation showing the subparts of an object and of how the subparts are connected, a physical representation showing photographs or drawings of the actual appearance of the object, a functional representation describing the objects in terms of functions and goals that the object must fulfill, etc. Surrounding the common knowledge-base is a set of tools that implement the different functions that the control system must perform. Figure 1.1 shows the basic system concept with the common knowledge-base and its surrounding tools. Some examples of general functions are control, monitoring, and diagnosis. Tools are also used to implement the user interfaces needed by different user groups. Operators, maintenance personel, and production engineers have different needs and requirements on the user interface with respect to what information that is presented and how the information is presented.

To make it possible to further develop and make conclusions about the concept, it is necessary to work against a "real" process. For this purpose, the UHT-process Steritherm from Alfa-Laval is used.

## Main purpose

The main purpose of this project is to examine the inner structure of the knowledge-base and how well G2 can implement this. In order to do this, a G2 model of a knowledge-based real-time control system will be developed. Since an actual Steritherm process is not physically available to the project, a separate numerical simulation model of Steritherm will be developed within G2. We will also examine how the problems associated with the Steritherm process can be solved with the G2 expert system shell. One problem is the burn-on that occurs in the heat exchangers. To prevent the burn-on from affecting the product quality, the production has to be stopped and the process

cleaned at regular time periods. This causes production losses. We will try to supervise the burn-on so it will cause minimal loss in the production. This is solved with expert system techniques. Also alarm analysis based on expert system techniques will be implemented.

## Outline of the report

Chapter 2 gives a brief introduction to knowledge-based systems and to the real-time expert system shell G2 used in the project. Chapter 3 contains an overview of the Steritherm process. Chapter 4 gives an overview of the system and how all the parts of the system fit together. In Chapter 5, 6, and 7, the main parts of the project are outlined. Chapter 5 describes the implementation and the inner structure of the simulation model. In Chapter 6, the implementation and the knowledge-base of the overall control system are described. Chapter 7 discusses G2's possibilities to represent and handle the knowledge used in this system. Finally conclusions are given in Chapter 8.

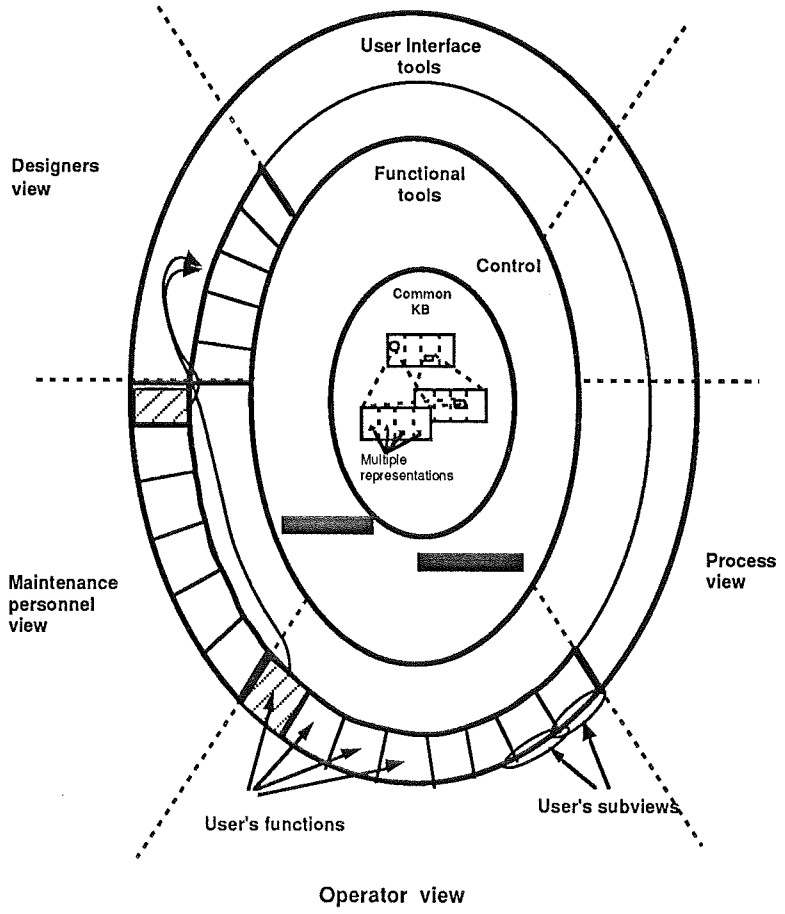



**Figure 1.1** Basic system concept

# 2 An overview of knowledge-based systems

This chapter gives a brief survey of the general concept of knowledge-based systems. It also gives an overview of G2.

## 2.1 Introduction

Knowledge-based systems or expert systems is an area of AI that has grown rapidly during the last few years. The basic definition of the term expert system is a program that solves problems within a specific, limited domain that normally would require human expertise. This is a wide and vague definition that also covers many traditional computer programs. A clear definition of an expert system is difficult to state. The situation instead is that an expert system more or less fulfills a number of different characteristics.

The most significant characteristic is the expert problem solving capacity within limited application domains and for problems where conventional programming techniques have not been successful. The reason why conventional techniques do not work is mainly that the problem lacks a clear analytic and/or algorithmical solution or that the existing algorithm is computationally intractable. The expert system instead tries to emulate the problem solving behavior of the human expert. This means that the system tries to represent and execute the expert's knowledge and reasoning strategy.

Another very common characteristic is that the domain knowledge is represented explicitly in an identifiable, separate part of the program. This so called knowledge-base is separated from the control strategy or inference engine that actually runs the program by operating on the knowledge. A system of this type is referred to as a knowledge-based system. The explicit knowledge representation gives an expert system a declarative nature in contrast to traditional procedural programming languages such as Pascal or Fortran where domain knowledge is expressed in the form of program statements. Knowledge representation, i.e. how individual facts are represented and how representations of individual facts are combined to a representation of the complete problem state, is a key issue in expert systems. Rules and object-oriented representation are often used tecniques wich are discussed in the following sections. Propositional logic or predicate logic, scripts, and procedures are other techniques. Well-developed explanation facilities and the possibility to reason with uncertainty are other characteristics of expert systems. Another aspect is the modularity that is provided through the explicit knowledge representation. The knowledge-base is built incrementally and can relatively easily be expanded with, e.g., new rules.

Most existing expert systems are implemented in some symbolical languages such as Lisp, Prolog or in some language implemented on top of those.

## 2.2 Object-oriented knowledge representation

To represent knowledge as objects with associated attributes is common in existing expert systems. The basis is the semantic network, which represents knowledge as a network of nodes. A node could represent the concept of objects, events, ideas, etc. Associative links represent relations among nodes.

### Frame systems

A frame system, which is a variation of the semantic network, consists of three different building blocks: frames, slots and facets. A frame is the equivalent of a node in a semantic network, i.e., it represents concepts of objects, events, ideas, etc. The slots or attributes describe the properties or attributes of a certain frame. In the same way facets describe the different slots.

Frames are often divided into two types: those which describe classes and those which describe individual instances. An important concept of semantic networks and frame systems is the inheritance of properties. Inheritance allows class frames which can pass their slots along to subclass frames and to instance frames.

### Object-oriented programming

The basic entity of object-oriented programming is the object which has a local state and a behavior. Objects are asked to perform operations by sending appropriate messages to them. Objects have associated procedures called methods that respond to the messages.

In the same way as in frame systems, objects are divided into classes and instances of classes. The classes build up a superclass-subclass hierarchy with inheritance. The inheritance is more focused on inheritance of behavior, i.e., of methods, than on inheritance of properties as is the case of frame systems.

## 2.3 Rule-based knowledge representation

Rules are the main knowledge representation method used in existing expert systems. This is reflected by the use of the name rule-based systems synonymously to expert system. Another name that is used is production systems where a production is the equivalent of a rule. Rules often look like:

**if** <antecedents> **then** <consequents>

or like

**if** <conditions> **then** <actions>

The main parts of a rule-based expert system are:

1. Database or working memory
2. Rulebase
3. Inference engine
4. User interface

The first two parts are commonly referred to as the knowledge-base. The database is used to represent facts about the application domain. The rulebase contains the rules of the system. It is sometimes partitioned into different rule-groups according to the different contexts. The left hand side of the rules typically consists either of patterns that should match the contents of the database or of predicates on the database that should be fulfilled. The inference engine applies the rules to the database according to some strategy. The dominating strategies are forward chaining and backward chaining. In some systems these two are combined. The two strategies will be further discussed below.

The power of rule-based systems is most evident for complex, ill-structured applications that lack efficient algorithmical solutions. The decomposition of the system into a number of relatively loosely coupled rules makes it suitable for problems that are decomposable into subproblems which have no fixed or apparent order.

### Forward chaining

In forward chaining systems, the left hand sides (LHS) of the rules are examined to see whether or not they are fulfilled. If so, the modifications to the database in the right hand side of the rules are executed and then the system examines the rules again. Forward chaining system are sometimes called data-driven systems. Most forward chaining systems use pattern matching to express when rules are applicable. The LHS of the rules contain patterns that must match the contents of the database for the rule to be fulfilled.

The reasoning is performed in what is called a recognize-act cycle that has three states. During the match state all rules that are fulfilled are collected into the conflict set together with the corresponding database elements. If rules with pattern matching variables are allowed, the same

rule can appear in the conflict set several times with different matching database elements. During the select phase, one rule is choosen for execution. If the conflict set contains more than one element, the conflict is resolved according to some conflict resolution strategy. During the act state the right hand side of the selected rule is executed.

### Backward chaining

Backward chaining systems are sometimes called goal-directed systems. A backward chainer tries to achieve a goal, or alternatively stated, to verify a hypothesis by trying to prove rules that confirm this hypothesis. A goal could, e.g., be expressed as the need to compute the value of a certain object attribute in the database, and a hypothesis could be expressed, e.g., as a certain value for an object attribute that needs to be verified. If the goal is not immediately available in the database, the backward chainer tries to find the rules with consequents that deduce the goal. A rule is selected and the antecedents of this rule become new goals that must be fulfilled. This causes new rules with these goals in their consequents to be selected an so on. The user is usually asked when a goal can not be directly proven and no rules are found with the goal in their consequents.

If a goal cannot be fulfilled the system backtracks and chooses another rule. The way in which the rules are selected and in which order the subgoals are analyzed determines the search strategy. During depth-first search the first applicable rule is chosen and its first antecedent immediately becomes the new subgoal. In breadth-first search strategy all the antecedents of the chosen rule are checked before a subgoal is selected. In best-first search the rule most likely to succeed, e.g., with the fewest antecedents, is selected first.

Backward chaining systems have traditionally well-developed explanation facilities. The explanation facilities are built into the inference engine and the explanations are generated automatically. The two standard types of explanation facilities are the "How?" and the "Why?" questions.

## 2.4 G2 as a Tool for real-time expert systems

This section contains an overview of G2, the expert system shell used in this project. G2 from Gensym Corp. is an expert system aimed at real-time, process industry applications. The main parts of G2 are: a knowledge-base, a real-time inference engine, a simulator, a development environment, an operator interface and optional interfaces to external on-line data servers.

## The knowledge-base

The knowledge-base consists of three different forms of knowledge: objects, rules and dynamic models. Objects are used to represent the different concepts of the application. Attributes describe the properties of a certain object. Attribute values may be constants, variables or other objects. The objects are organised into a class hierarchy with single inheritance. Objects are represented by graphical icons as shown in Figure 2.1. Relations between objects are represented by connection objects. Usually, objects are used to represent the physical components in an application with the connections representing physical connections such as pipes or wires. It is also possible to have objects that represent abstract concepts and connections that represent general relations among objects. Variables are a special type of objects for representing parameters whose values vary over time. Variables are either quantitative, symbolical, logical or textual and have validity intervals indicating the length of time the value remains valid after having been updated. It is possible to indicate that the validity should be computed from the validity intervals of the variables from which the value is inferred. Other variable attributes determine from where the variable receives its value (e.g. the simulator , the inference engine, or a data server), and whether a history should be kept for the variable or not. G2 contains built-in functions for referencing past variable values and for the usual statistical operations on time series such as mean value, standard deviation, rate of change, maximum, minimum, etc.

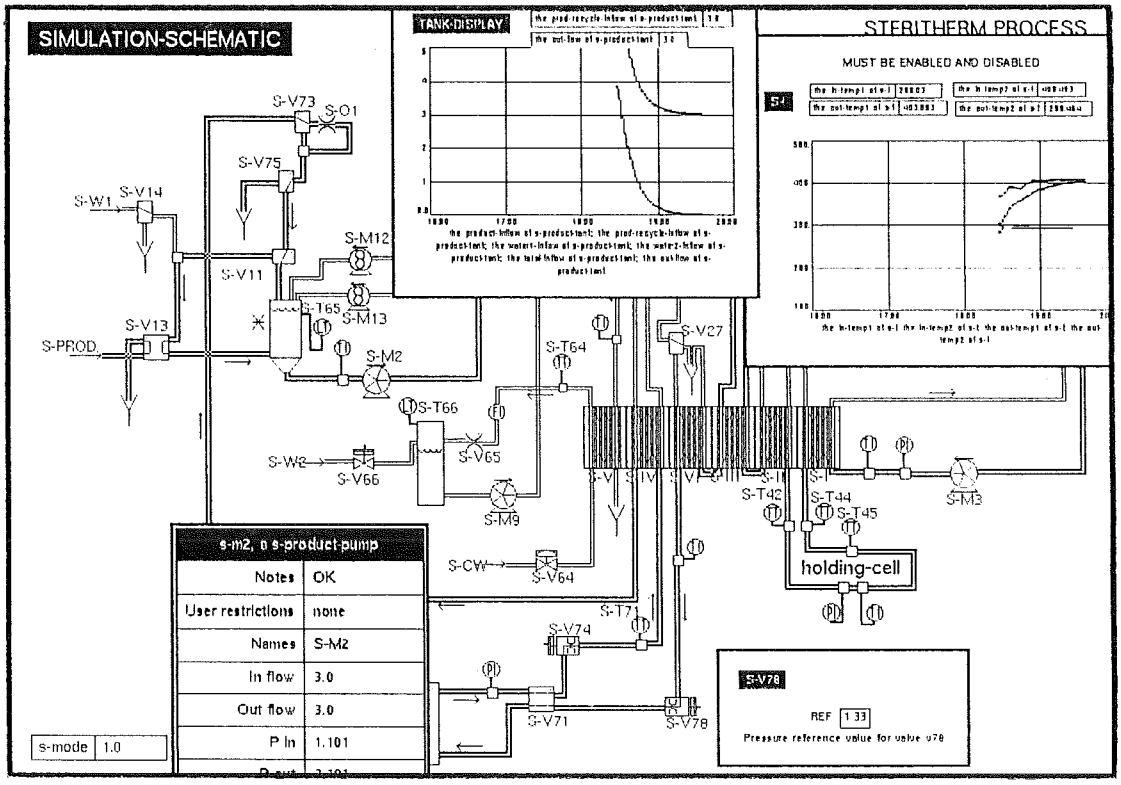



**Figure 2.1** A typical G2 display screen on Symbolics

G2 rules are used to encapsulate an expert's heuristic knowledge of what to conclude from conditions and how to respond to them. Five different types of rules exist. Of them, four are different forms of "if conditions then actions" rules. The last type is the "whenever" rule which allows asynchronous rule firing as soon as a variable receives a new value or fails to receive a value within a specified time-out interval. The rule conditions contain references to objects and their attributes in a natural language style syntax. Objects can also be referenced through connections with other objects. G2 supports generic rules that apply to all instances of a class. The G2 rule actions make it possible to conclude new values for variables, send alert messages, hide and show workspaces, move, rotate, and change colour of icons, etc. G2 rules can be grouped together and associated with a specific object, a class of objects, or a user-defined category. This gives a flexible way of partitioning the rule-base. The following is an example of a generic G2 rule that applies to all water tanks:

**for any** water-tank
  **if** the level of the water-tank < 5 feet **and**
    the level-sensor connected to the water-tank is working
  **then** conclude that the water-tank is empty **and**
    inform the operator that "[the name of the water-tank] is empty"

Dynamic models are used to simulate the values of variables. The models are in the form of first-order difference and differential equations. The models can be specific to a certain variable or apply to all instances of the variable class.

### The real-time inference engine

The real-time inference engine initiates activity based on the knowledge contained in the knowledge-base, simulated values, and values received from sensors or other external sources. Apart from the usual backward and forward chaining rule invocation, rules can also be invoked explicitly in several ways. First, a rule can be scanned regularly. Second, by a focus statement all rules associated with a certain focal object or focal class can be invoked. Third, by an invoke statement all rules belonging to a user defined category, like safety or startup, can be invoked. The scanning of a few vital rules in combination with focusing of attention is meant to represent the way human operators monitor a plant. It is also an important way to reduce the computational burden on the system. Regular scanning of rules and thus updating of information in combination with variables with a time-limited validity gives a partial solution to the problem of non-monotonic, time-dependent reasoning. The inference engine automatically sends out requests for sensor variables that have become non-valid and waits for new values without halting the system. Priorities can be associated with rules.

G2 has a built in simulator which can provide simulated values for variables. The simulator is intended to be used both during development for testing the knowledge-base, and in parallel during on-line operation. In the latter case, the simulator could be used for estimation of signals that are not measured. Each first-order differential equation is integrated individually with individual and user-defined step-sizes.

### The environment

G2 has a nice graphics-based development environment with windows (called workspaces), pop-up menus, and mouse interaction. Workspaces can be hierarchically organized through the possibility to associate a subworkspace with an object. Input of rules and other textual information is performed through a structured English-style grammar editor. Facilities for browsing through the knowledge-base exist. The operator interface contains variable displays such as graphs, meter, read-out tables, etc., and operator controls in the form of different types of buttons and sliders for changing variables and executing rule actions.

The data servers are interfaces to either conventional control systems or other signal sources such as databases.

G2 is implemented in Common Lisp and runs on Sun, HP, DEC Vaxstation, TI Explorer, Microexplorer, Symbolics (which this project uses), Compaq 386, and Mac II. For most machines, 16 Mb RAM memory is required.

# 3 The Steritherm process

This chapter will give an overview of the Steritherm process which is used in this project.

## 3.1 Introduction

The Steritherm process is a food-production process used to sterilize milk and other similar products. The sterilization is done with a technique called UHT (Ultra High Temperature) treatment. This means that the product is heated up to a temperature of 137 °C for a short period of time. This will kill all existing micro-organisms. UHT treatment allows the products to be stored in room temperature for long periods of time.

The process runs continuously in a closed system to protect the product from new micro-organisms in the air. The product passes through heating and cooling very fast. Aseptic packing of the product is an integrated part of the process.

## 3.2 The process

The Steritherm process has been developed by Alfa-Laval, a Swedish food engineering company. The configuration may differ a bit from process to process. Here we will describe the configuration of the Steritherm process used in this project.

The process consists of one balance tank for product and one balance tank for heating water, heat exchangers for cooling and heating the product and the water, a steam injector for heating up the water, a packing machine, pumps, and a number of valves. See Figure 3.1 for a detailed scheme of the plant.

The heat exchangers are connected in a regenerative way to save energy. This means that before the product is heated up to the sterilization temperature, it is preheated by product that already has been sterilized. This in turn, means that the product when it is preheated also cools the sterilized product. The same procedure is repeated for the heating water.

The product comes into the product tank from a silo tank outside the process. From there the product is pumped through a heat exchanger section for preheating to a heat exchanger section where the sterilization temperature (137 °C) is reached. After that, it is cooled in three heat exchanger sections and finally it reaches the packing machine where it is aseptically packed. Some amount of product is recirculated back to the product tank. The reason for this is that there must be a certain over-pressure in the packing machine.

The water from the water tank is pumped through one heat exchanger section for preheating and then to the steam injector where steam is injected directly to the water,increasing the water

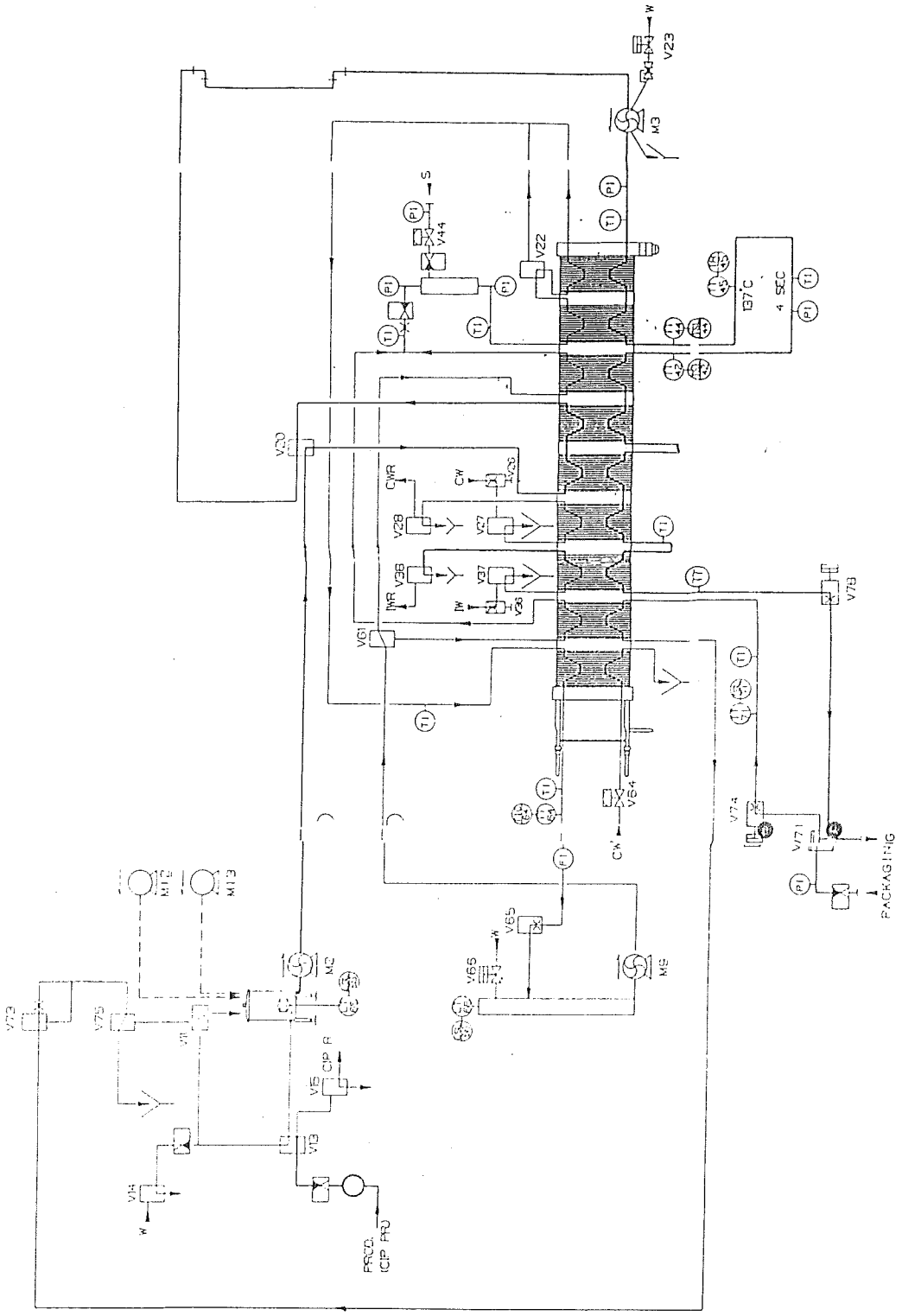



**Figure 3.1** The schematic of the Steritherm process

temperature considerably. After the steam injector, the water heats up the product to the sterilization temperature (137 °C) in another heat exchanger section. The water is then cooled and recirculated back to the water tank. There are also optional heat exchanger sections for additional cooling. In the schematic in Fig. 3.1 there are two extra sections for cooling but in the process used in this project, only one extra cooling section is used.

There are several different valves in the process. The valve v44 regulates the temperature of the product in the holding cell and the valve v64 regulates the ice water that cools the recirculated warm water. These two valves are controlled by two PID regulators. The constant pressure valves v78 and v74 control the pressure in the holding cell and in the packing machine. This means they indirectly control the flow of product into and from the packing machine. The other valves are merely used to control the direction of the flow in the process depending on which phase the process is running in (see the section below).

## 3.3 Operation phases

The Steritherm process operates in a number of different phases where the production phase is the most important.This section will describe the main phases of the process. In each of these phases there exists subphases which are described in Appendix A.

### Sterilization phase

In the sterilization phase the plant is sterilized. Water is circulated in the product line and heated-up to 137 °C.

### Sterile water phase

In the sterile water phase water is circulated in the product line and the plant is ready to begin the production.

### Production phase

In the production phase the UHT-treated product is produced.Product is circulated from the product tank through heat exchangers to the packing machine and a small part is recirculated back to the tank.

### AIC phase

In this phase, called Aseptic Intermediate Cleaning, the plant is cleaned by circulating a mixture of

water and caustic through the process. After the plant has been cleaned it returns to the sterile water phase and is ready for production again.

### CIP phase

This phase, called Cleaning In Place, is more exhaustive than the AIC cleaning. In this process a mixture of water and acid, and a mixture of water and caustic is circulated several times successively. After this the plant has to be resterilized (which was not necessary in the AIC phase) before it goes to sterile water and is ready for production.

# 4 The structure of the system

The system consists of two major parts, the dynamic simulation model and the control system.

The simulation model is mainly used to replace the "real" Steritherm process. This will allow faster development and verification of the control system.

The control system is the major and most interesting part of the project. It consists among others of hierarchical views from the dairy level down to components level, control blocks with PID-regulators, alarm and fault analysis, sequences for the different phases and rules for burn-on supervision.

The connection between the control system and simulation model is only made through the sensors used in the process and through the control signals from the control system. Figure 4.1 show this connection. This gives a clean interface between the control system and the model. It would not involve too much work to replace the simulation model with an interface to a real Steritherm process.

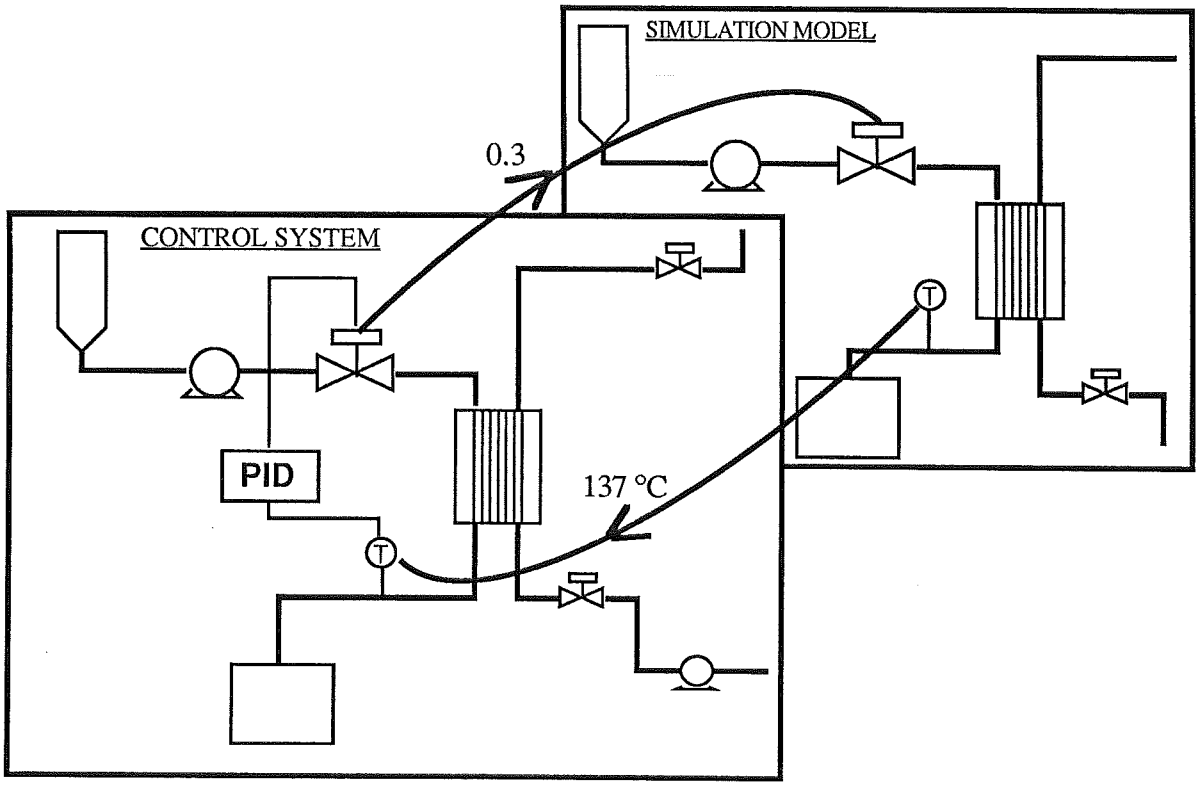



**Figure 4.1** The principal connection between the control system and the simulation model

# 5 The Simulation model

This chapter describes the simulation model for the Steritherm process. After a brief introduction we give a description of all the different objects, their attributes and all variables used in the model. Then we describe the simulation model for each object and finally how the overall simulation of the process is accomplished.

The schematic of the simulation model is built on a separate G2-workspace without any hierarchical levels. There are several top-level workspaces that keep information regarding the simulation model.These are the: simulation-schematic, simulation-object-definitions, simulation-variable-definitions, and simulation-variables.

The most interesting variables in the simulation are the pressures, flows, and temperatures in the valves, pumps and heat exchangers and the level in the balance tanks. The basis for the simulation models of pumps, valves and heat exchangers has been taken from the Department of Automatic Control (Åström, 1974; 1989).

The main operation phases of Steritherm, i.e. production, sterilization, etc, are made up of several subphases that sometimes can differ a bit from plant to plant. The activation chart that shows the phases that will be simulated in this model is described in Appendix A. There are unfortunately a few phases that are difficult to simulate in a realistic way in G2. In the phase COOLING1-113, COOLING2-114, RINSING-158 and ACID DOS-169 the valves v20 and v61

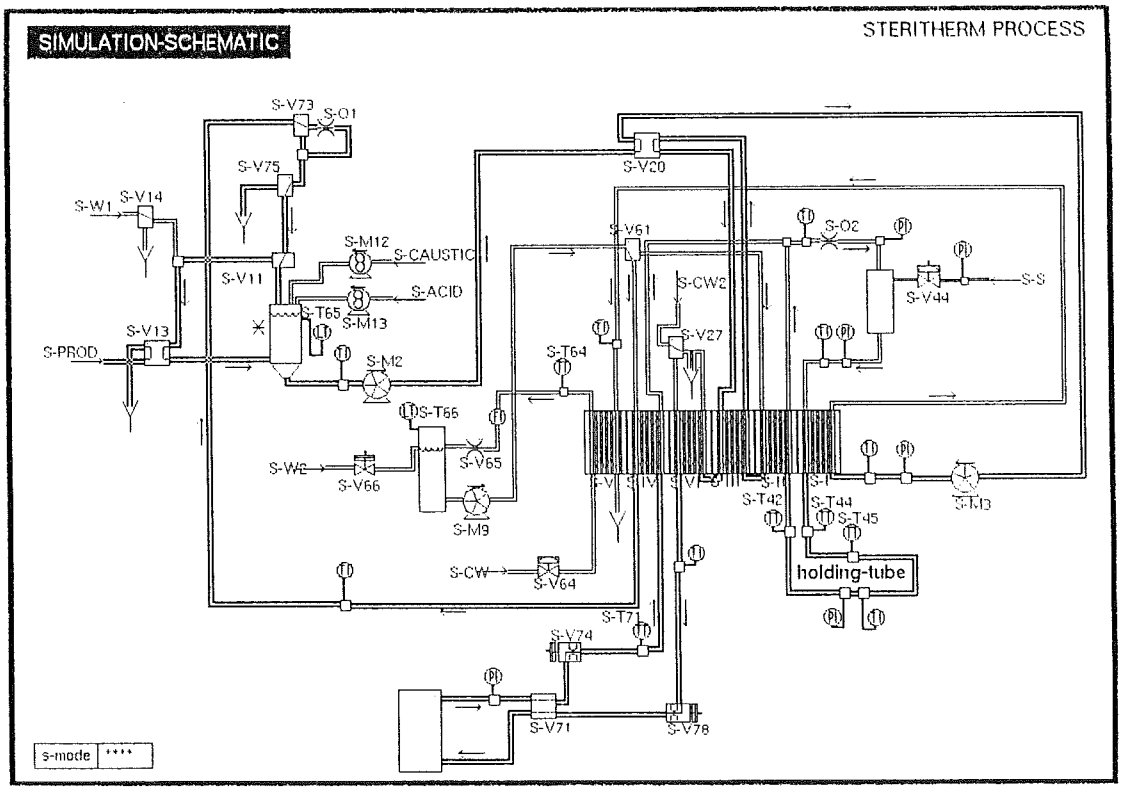



**Figure 5.1** The simulation schematic in G2

are slowly changing position. These phases will be simulated as if the valves have already reached their end positions. In the phases PROD IN-118 and PROD OUT-121 there are both product and sterile water in the plant and this will be simulated as if there is only one of them.

The burn-on in the heat exchangers will be simulated in two ways. First by reducing the heat transfer coefficient and second by increasing the pressure drop in the heat exchangers.The pressure drop is increased by successivly decreasing the flow area through the heat exchanger .

Since the temperature simulation is dynamic, the burn-on will be simulated in a rather realistic way. This is an advantage since burn-on supervision is an important part of the project.

## 5.1 The structure of the simulation model

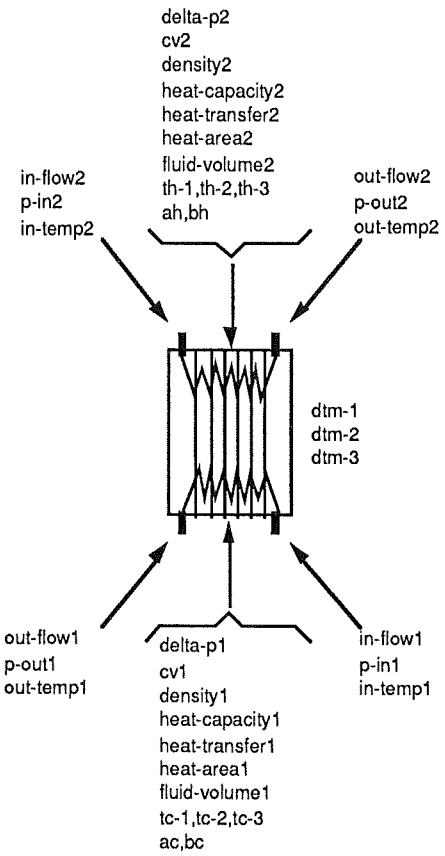
As a base for the simulation model built in G2 we have used a simplified schematic of a Steritherm process. This schematic is shown in Figure 3.1 in Chapter 3. The corresponding simulation schematic in G2 is shown in Figure 5.1. The objects that shall be part of the Steritherm model are pumps, valves, heat exchangers, tanks, a steam injector, regulator blocks, sensors, pipes, sources, and finally a packing machine. The objects are defined hierarcically and can be found on the top level workspace simulation-object-definitions. On the highest level there are the classes s-process-equipment, s-indicators and s-connections. The class s-process-equipment is the most interesting in the simulation model. This class has the subclasses s-plate-heat-exchanger, s-pump, s-valve, s-balance-tank, s-source-and-sink, and s-steam-injector. All of the classes and objects connected to the simulation model have the prefix s to distinguish them from the declarations of the corresponding classes in the control system. In the following, we will discuss the different object classes, variable definitions, and global constants and variables used in the simulation model.

### S-process-equipment

In this section we will explain the characteristics of all subclasses of s-process-equipment.

### S-plate-heat-exchanger

The icon of this class and its attributes are shown in the figure below. The suffixes 1 and 2 are only used to differ the two sides of a heat exchanger. The suffixes 1, 2, and 3 in dtm, th, and tc are used to split the heat exchanger into three sections which is necessary for the temperature equations in the simulation (see Section 5.2 about the heat exchangers).



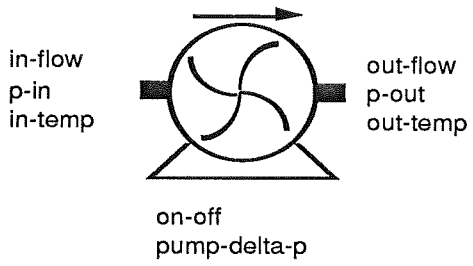

The attributes concerning temperature, pressure, flow, density, heat capacity, heat-area and fluid-volume are rather self explanatory. In the rest of this section we will only explain the attributes that may be difficult to understand.

The cv attributes correspond to the flow resistance of the heat exchanger. The delta-p attributes are the pressure drop across the heat exchanger. The heat-transfer attributes are the total heat transfer coefficients from one side to the other. The attributes th, tc, dtm, ah, ac, bh and bc are used in the temperature equations for the heat exchangers and will be explained later in this chapter (see Section 5.2 about the heat exchangers). The simulation equations for this class are generic and can be found in its subworkspace.

There are three subclasses of s-plate-heat-exchanger. This is because some heat exchangers have water connections at both sides, some have product connections and some have one of each type of connection. These subclasses have no attributes of their own but they inherit the attributes of the superior class.

### S-pump

The principal icon of a pump and its attributes are shown in the following figure.



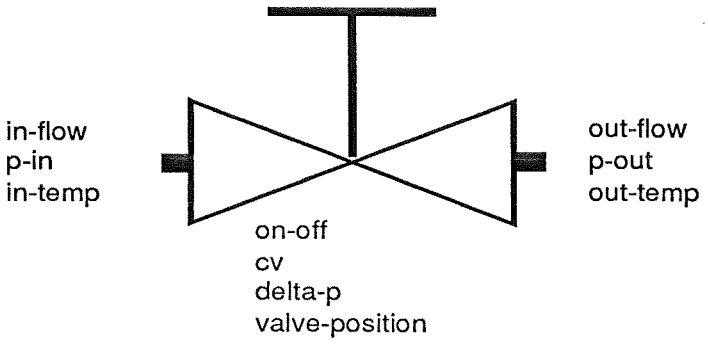

The attribute pump-delta-p is the increase in pressure across the pump. The attribute on-off indicates whether the pump is running or not.

There are two sublasses of s-pump: s-centrifugal-pump and s-displacment-pump. The s-displacement-pump has the additional attribibute flow-ref that states the flow that the pump will generate. S-centrifugal-pump has in turn two subclasses because it must handle both water connections and product connections.

### S-valve

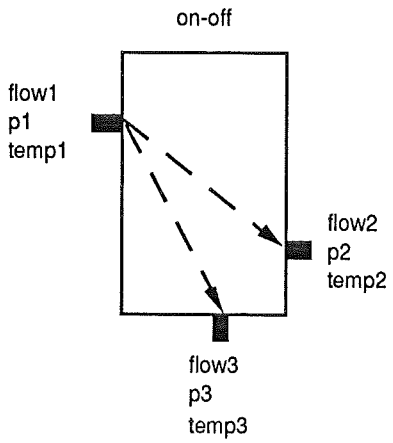
S-valve has three subclasses: s-1-to-1-valve, s-1-to-2-valve, and s-2-to-2-valve. S-1-to-1-valve is a valve that has one inport and one outport. The flow through the valve can either be regulated manually or automatically. S-1-to-2-valve is a valve with one inport and two outports. It is only used to control the direction of the flow. The incoming flow is directed to one of the two outports. S-2-to-2-valve is a valve with two inports and two outports. This valve also controls the direction of the flow. The two outports can be connected with either one of the two inports.

The principal icon of s-1-to-1-valve and its attributes are shown in the figure below.



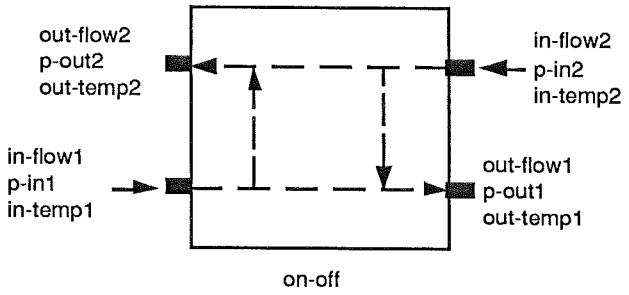

The on-off attribute indicates whether the valve is active or not. When it is inactive the valve is fully opened and acts like a pipe. The cv value corresponds to the flow resistance in the valve and delta-p is the pressure drop across the valve. The valve-position is a number between zero and one that indicates how much the valve is opened.

The icon of s-1-to-2-valve and its attributes are shown below.




The suffix 1, 2 and 3 in the attributes refer to one of the three ports of the valve. Suffix 1 correspond to the inport and suffix 2 and 3 to the outports. The attribute on-off indicates the direction of flow.When on-off is on then the flow1 is directed to flow2 otherwise it is directed to flow3. The same goes for the other attributes.

The icon of s-2-to-2-valve and its attributes are shown in the following figure.




Since there are two inports and two outports we use suffix 1 and 2 to separate them. The attributes on-off indicates the direction of flow in the valve. When on-off is on the in-flow1 is directed to out-flow1 and in-flow2 is directed to out-flow2. Otherwise the in-flow1 is directed to out-flow2 and in-flow2 to out-flow1.

The s-1-to-1-valve class has the three subclasses s-constant-flow-valve, s-constant-pressure-valve, and s-regulator-valve.

S-constant-flow-valve is a regulating valve that tries to maintain a constant flow through the valve. This valve has the attribute flow-ref which is the reference flow. Since it has to be connected to both water and product lines the class has the two subclasses s-const-flow-valve-for-water and and s-const-flow-valve-for-product.

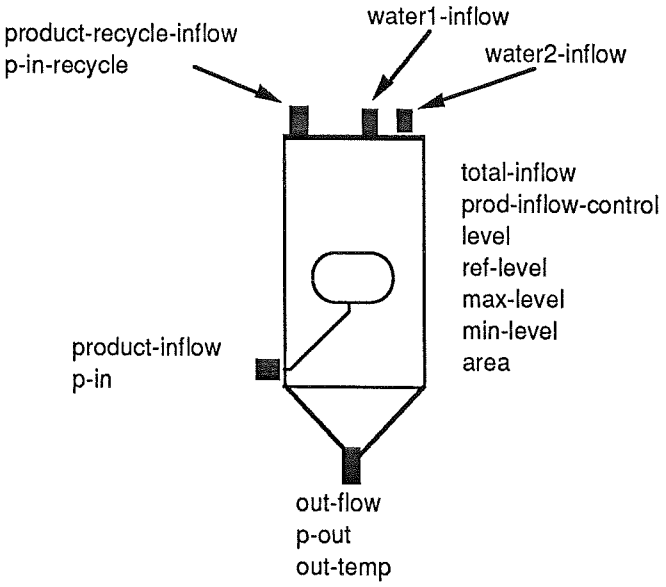
S-constant-pressure-valve is a regulating valve that tries to maintain a constant pressure at the inport. It has the attribute p-in-ref which is the reference pressure.

S-regulator-valve is a valve where the throttle that controls the flow through the valve is controlled by an external regulator. It has no attributes of its own and three subclasses which again is a result of the different connections that the valve must handle.

### S-balance-tank

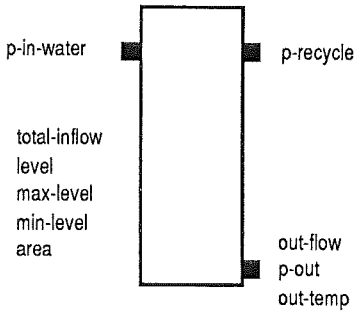
This class describes the two tanks that are used in the process. This class has no attribute but it has the two subclasses s-balance-tank-for-product, that describes the tank in the product line and s-balance-tank-for-hotwater, that describes the tank used in the water line.

The icon for s-balance-tank-for-product and its attributes are shown in the figure below.




The product-inflow is the flow of product into the tank. The prod-recycle-inflow is the recycled flow from the packing machine. The water1-inflow and water2-inflow refer to the flow of washing liquid from the two displacement pumps connected to the cleaning tanks. The prod-inflow-control corresponds to the float that controls the cross area of the inlet port of the product. When the level is higher than max-level the inlet port is closed (i.e. prod-inflow-control is zero) and when the level is lower than min-level the inlet port is maximally opened (i.e. prod-inflow-control is one). The actual inflow of the inlet port is the product between the flow given by product-inflow and the prod-inflow-control. The total-inflow is the sum of all flows into the tank. The out-flow, p-out and out-temp correspond to the flow, pressure and temperature at the outlet port of the tank. The p-in is the pressure at the inlet port for the product. P-in-recycle is the pressure at the port for the recycled flow from the packing machine. The ref-level is the reference value for the level in the tank. The area attribute is the cross section area of the tank.

S-balance-tank-for-hotwater has the attributes shown in the following figure.



These attributes are similar to the product tank attributes discussed above.

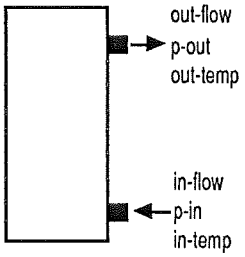
### S-source-and-sink

The only attribute in this class is on-off that tells whether the object is active or not. The class has the two subclasses s-source and s-sink-source.

S-source is a class that supplies steam and different kind of liquids to the model. It has the attributes out-flow, p-out and out-temp. It has three subclasses for connecting to different pipes.

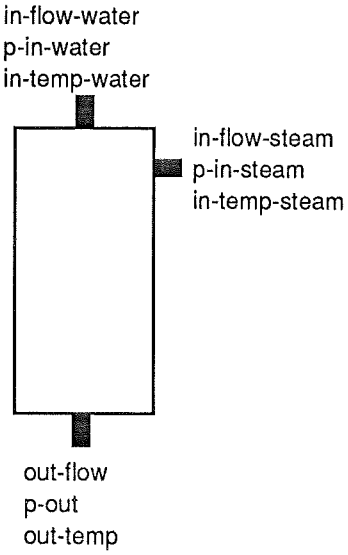
The class s-sink-source is a packing machine and the icon and its attributes are shown below. The reason why the packing machine can be seen as both a sink and a source is the follwing. The product is flowing into the packing after the UHT-treatment. Depending on the pressure in the packing machine, it will produce a flow from the packing machine.

The packing machine has an outport for recycled product flow.




### S-steam-injector

This class describes the steam injector in the process. A steam injector is used to heat up water by injecting steam directly into the water. The icon of this class and its attributes are shown below.




The steam suffix corresponds to the steam inport and the water suffix corresponds to the water inport.

### S-indicator

Under this class all different sensors used in the simulation model are declared. It would have been more accurate if this class had been a subclass to s-process-equipment but because all sensor variables must have the superior G2 class sensor, it must be separated from the s-process-equipment class.

S-indicator has the subclasses s-pressure-indicator, s-differential-pressure-transmitter, s-level-transmitter, s-temperature-indicator, s-flow-indicator, and s-temperature-transmitter. These subclasses correspond to the different kinds of sensors used in the simulation model. These indicators make up the "bridge" between the simulation model and the control system.

### S-connections

Under this class all different pipes and wires used in the simulation model are declared. There are 4 different kinds of subclasses.These are product-line,water-line,steam-line,and instrument-signal.

The classes have no attributes.It is not necessary because the pipes are simulated as ideal pipes with no losses. The instrument-signal connection is only used by the sensors.

As mentioned before, the Steritherm process has many different phases. In different phases, different kinds of liquids will circulate in the same pipe. This is a problem when constructing the Steritherm schematic. There is two solutions to this problem. First, one can use only one kind of pipes in the process but this gives a very poor survey over the process. Second, one can choose a particular phase and choose the pipe classes after the liquids that circulates in the different sections in the process in this phase. This is the solution that has been used. It gives a clearer view over the process. The apperance of the pipes has been chosen after the production phase.

When defining objects, stubs are declared to be able to interconnect the objects. One can only connect to stubs that are of the same class. They can not be mixed. For example, connecting a valve with stubs declared as product-line stubs into a water-line is not possible. This problem results as you probably already have noticed, in many unnecessary subclasses.

### Variable classes

In the simulation model there are three different kinds of variable classes. These are fast-sim-var, fast-sim-var-with-history, and slow-sim-var; and, they are found on the top level workspace simulation-variable-definitions.

The data server attribute for these classes are the G2-simulator and they are all subclasses of

the G2-variable class quantitative-variables.

The fast-sim-var and fast-sim-var-with-history classes have a validity interval of two seconds while the slow-sim-var has an indefinite validity interval. Fast-sim-var-with-history stores a history of old variable values for three minutes. The fast-sim-var and fast-sim-var-with-history variable classes are used by variables whose values are changed dynamically while the simulation runs. Almost all attributes in the objects are of these types. For exampe all flow, pressure and temperature attributes are declared as one of these two variable classes. The slow-sim-var class is used for variables which are very seldom changed. These variables get their values from the control system using a "set" action. If a variable is shown on a graph, it must be of the class fast-sim-var-with-history.

### Global constants and variables

There are two global constants used in the simulation model. These constants are really variables but they are used as constants (G2 does not have the datatype constant). The global constants are dt and p-atmosphere. The time step Dt, is used in the tanks when computing the simulated values for the levels.

The global variables used in the simulation model are density, heat-capacity, temp-burn-on-characteristic, pressure-burn-on-characteristic, burn-on-simulation, burn-on-AIC-factor, s-mode and mode. The density and the heat-capacity are characteristics of the product that are currently used in the Steritherm process. The temp-burn-on-characteristic, the pressure-burn-on-characteristic, burn-on-simulation and burn-on-AIC-factor are characteristics that deals with the burn-on. These will be discussed in more detail in Section 5.3 which is about the burn-on simulation. S-mode keeps track of the current "internal" phase of the Steritherm simulation. The simulation part in G2 sometimes need two or more "internal" phases when simulating one real phase in the Steritherm process to make the simulation work properly. The "internal" phase should not be confused with the real phase of the Steritherm process. The real phases are shown in the activation chart in Appendix A. Because of the need to record every time s-mode recieves a value, the variable mode is used instead. Mode is an inference variable and can therefore be used in a "whenever rule" that is tested every time mode gets a new value. S-mode is a simulation variable and may not be used in "whenever rules". However s-mode is necessary, since the simulator can only refer to simulation variables. It is the control system that gives mode a value depending on the current phase of the process via a "conclude rule" and mode in turn "sets" s-mode to the same value.

## 5.2 Implementation of the simulation model

To implement the simulation equations for the Steritherm process, some simplifications and assumptions concerning the different objects' physical properties have to be made. This is necessary for many reasons. One important reason is that the equations would be too big and complex otherwise. The difficulties with the simulation model also depend on the several different phases of the Steritherm process and the fact that in each phase different valves and other process components are active.

The Steritherm process consists of two main lines. The water line is used for heating and cooliong of the product. This water line is indicated in Fig. 5.2. The product line is where the product flows during the production phase. In Figure 5.3 the product line is filled and marked with arrows. The lines contain several valves that have different positions in the different phases. This means that the flow goes through different objects depending on the current position of the valves. Furthermore, there are some valves that sometimes are active and control the flow or pressure in the lines, and sometimes are inactive and act as if they were not there. These are some of the reasons why the simulation equations are difficult to write.

When simulating flow and pressure some problems arise. To calculate the flow one has to look at the process from a global point of view. One must know all the different objects that are involved in a particular line (compare with resistances and currents in an electrical circuit), which objects that increase the pressure, and which objects that decrease the pressure and so on. To be

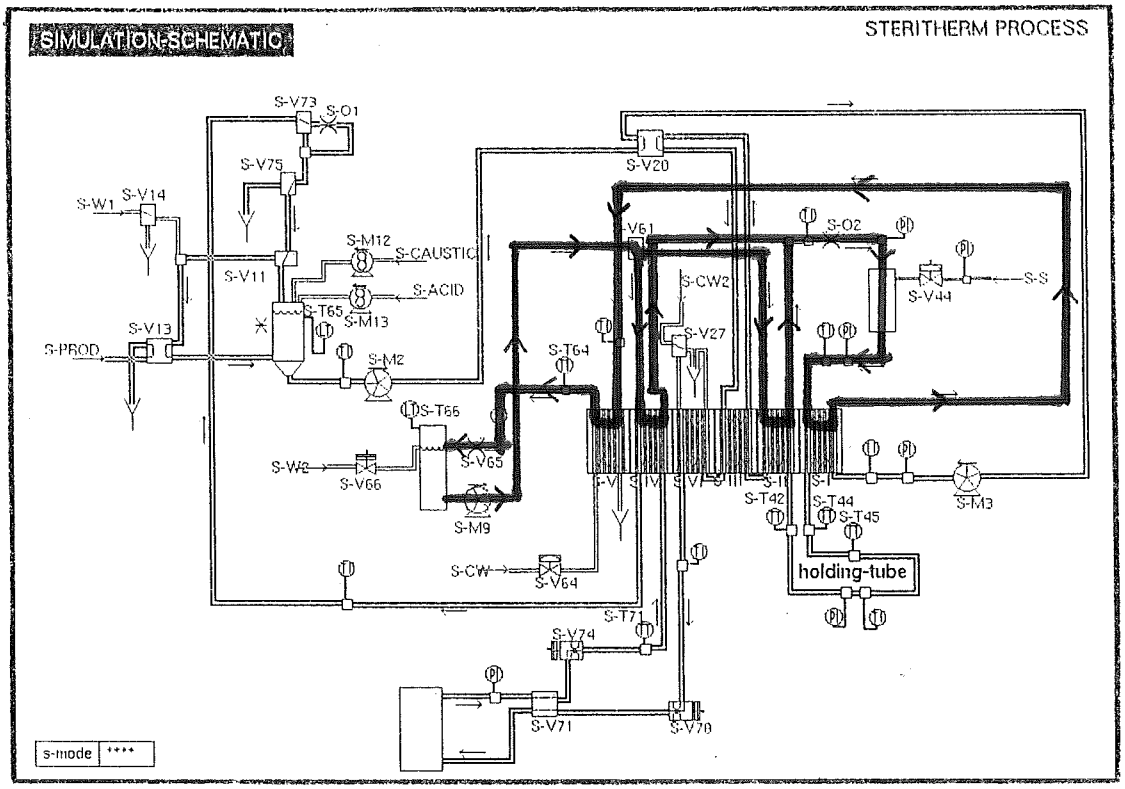



**Figure 5.2** The water line in the simulation schematic

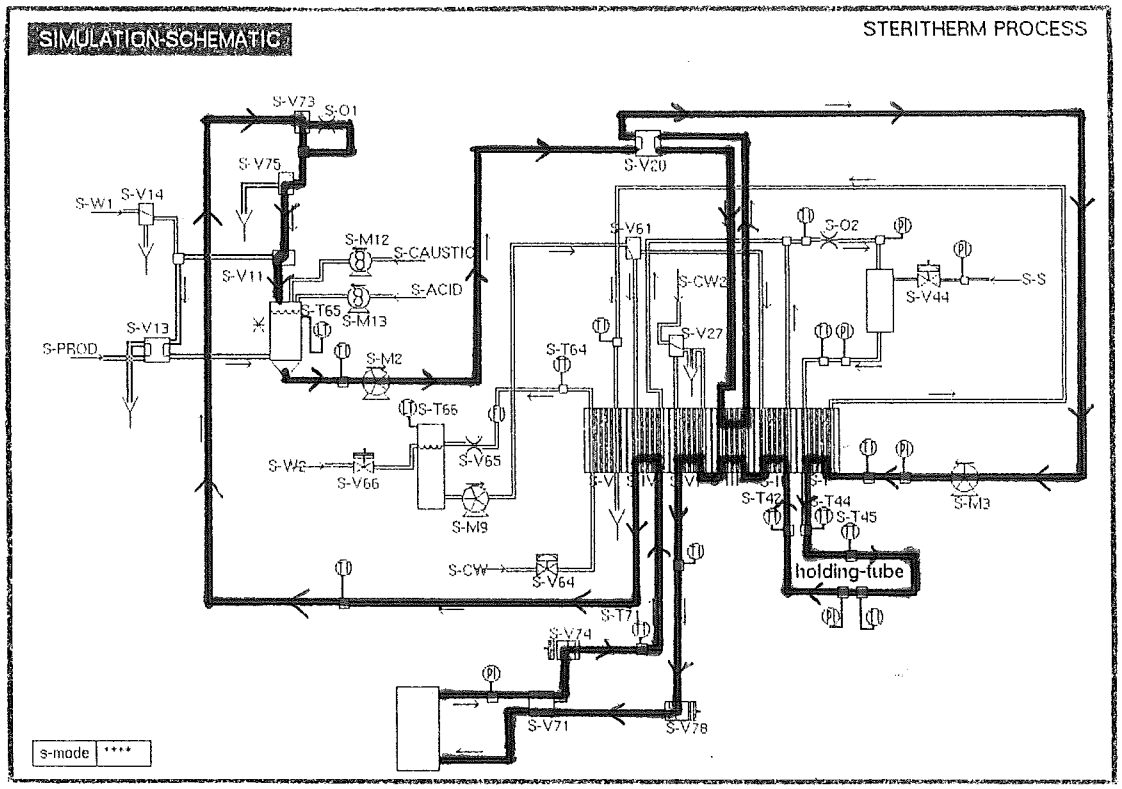



**Figure 5.3** The production line in the simulation schematic

able to handle these problems, we have implemented a few global functions that calculate the flow in the different lines and during the different phases. By computing the flow (through a call to a global function) at the first object in a line, for example the out-flow attribute in the product tank in the product line, the flow can be propagated together with the calculated pressure and temperature through every object in the line. With this approach, the components in the lines can easily calculate their pressure drops which also means that the pressure at different points in the process can be calculated.

Following this chapter we will give a brief description of the different simulation models for the objects used in the process.

### Tanks

There are two different tanks in the Steritherm process, a production tank and a water tank. Their functions (from a simulation point of view) are similar with the exception that the production tank has a float that controls the level in the tank. Both tanks are atmospheric.

The change in the level L is given by

$$\frac{dL}{dt} = (q_{in} - q_{out})/A$$

where

$q_{in}$ = The total inflow of the tank
$q_{out}$ = The outflow of the tank
A = The cross area of the tank

The pressure at the outlet is given by

$$P_{out} = 9.81*\rho*level + P_{atm}$$

The out flow is calculated by one of the global functions (see Section 5.3 about the flow and pressures simulation). The temperature of the liquid is given by adding the temperature of every inflowing liquid, multiplying each with its flow and dividing this result with the total inflow. We assume here a homogenous and direct mixing of the incoming liquids.

### Pumps

Two different kinds of pumps exist in the process, centrifugal pumps and displacement pumps. The centrifugal pumps have a predefined increase of pressure which is assumed to be flow independent. The displacement pumps are modelled as constant flow sources. Neither of these pumps affect the temperature of the liquid.

### Valves

There are three different kinds of valves in the Steritherm process: regulator valves with one inport and one outport, valves with one inport and two outports, and valves with two inports, and two outports. The latter two of the three types are only used to control the flow in one or another direction. They only propagate pressures, flows and temperatures.

There are three different kinds of regulator valves. They have in common that they are modelled as ideal valves with a pressure drop that depends on the flow through the valves. The model for the pressure drop is given by

$$P_{in} - P_{out} = \rho q^2 / C_v^2$$

where

$P_{in}$ = the pressure at the inport
$P_{out}$ = the pressure at the outport
q = the flow
$\rho$ = the density
$C_v$ = the flow coefficient

The flow coefficient corresponds to the flow resistance, R, through a valve. The flow coefficient is given by:

$$C_v = \frac{1}{\sqrt{R}}$$

By changing the cross area of a valve you change the flow coefficient (i.e the flow resistance) and thereby the flow and pressure drop across the valve.

The first type of regulator valves are controlled by a PID controller that regulates the valve position (i.e. flow coefficient based on temperature measurements).

The second type is a constant pressure valve that aims to keep the pressure at the inlet constant at a predefined reference pressure. There is a simple P-regulator implemented that controls the valve position of the valve and thereby the pressure at the inport.

The last of the three regulator valves is the constant flow valve that aims to keep the flow through the valve constant due to a predefined reference flow. This valve works just as the preceeding valve with the only exception that the reference value for the P-regulator is a flow instead of a pressure.

### Heat exchangers

With respect to pressures and flows the heat exchanger is modelled as a valve, i.e. the flow through the heat exchanger is mainly dependent on the pressure drop across the heat exchanger and a constant, the CV-number. The simulation formula for the pressure drop is identical to the pressure drop of a valve, that is

$$P_{in} - P_{out} = \rho q^2 / C_v^2$$

The simulation formulas for the temperature in the heat exchangers are a bit more complicated. We have used a dynamical model for counterflow heat exchangers by Åström (1989) which is described by the following equations:

$$\frac{\partial T_h}{\partial t} + a_h \frac{\partial T_h}{\partial (x_h/l)} + a_h b_h (T_h - T_c) = 0$$

$$\frac{\partial T_c}{\partial t} + a_c \frac{\partial T_c}{\partial (x_c/l)} + a_c b_c (T_c - T_h) = 0$$

where

$a = q/V$ inverse residence time

$b = \frac{kA}{q\rho c}$ number of heat transfer units

$q$ = flow $\left[m^3/s\right]$

$V$ = fluid volume

$A$ = heat transfer surface

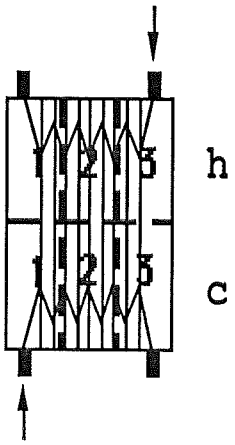
$k$ = total heat transfer coefficient

$\rho$ = density $\left[kg/m^3\right]$

$c$ = specific heat capacity

The indices c and h correspond to the cold and the hot side of the heat exchanger.

To simulate the model with reasonable effort the partial differential equation is approximated by a set of ordinary differential equations by discretizing the space variable. The discretizing is done here by segmenting both the hot and cold side of the heat exchanger into three equally sized parts. The heat exchanger below shows the hot and cold side and how the segmentation is done.




The approximation now becomes

$$\frac{dT_{hi}}{dT} = a_{hi} (3 (T_{h(i+1)} - T_{hi}) - b_{hi} \Delta T_{mi})$$

$$\frac{dT_{ci}}{dT} = a_{ci} (-3 (T_{ci} - T_{c(i-1)}) + b_{ci} \Delta T_{mi})$$

$$i = 1,2,3$$

where

$$\Delta T_{mi} = \frac{\Delta T_{chi} - \Delta T_{ch(i-1)}}{\ln(\Delta T_{chi} / \Delta T_{ch(i-1)})}$$

$$\Delta T_{chi} = T_{h(i+1)} - T_{ci}$$

Finally, the last part is further approximated as

$$\Delta T_{mi} = 0.5(\Delta T_{chi} - \Delta T_{ch(i-1)})$$

The index i refers to one of the three segments that the heat exchanger is divided into. The reason for this approximation is that it decreases the load on the simulator and still gives acceptable simulation results.

When implementing the equation in G2 we could not use exactly the same notation as above (due to a restriction when writing simulation formulas in G2). The notation used is

$a_{hi} = Ah$
$a_{ci} = Ac$
$b_{hi} = Bh$
$b_{ci} = Bc$
$\Delta T_{mi} = Dtmi$
$T_{hi} = Thi$
$T_{ci} = Tci$
$i = 1,2,3$

Notice that there are no indices in the first four lines. This is only because the volume, the heat transfer surface, the flow and the heat transfer coefficient are the same for all segments of one side.

It may seem to be a very rough approximation to divide the heat exchangers in to only three parts, but it proved to be sufficient to give a realistic behaviour of the temperature in the heat exchangers. Reducing the load on the simulator is another reason for having as few segments as possible.

When the flow in one side of the heat exchanger is zero, the temperature on the other side is the same in all three segments, i.e. the heat exchanger is inactive when simulating the temperature.

When tuning the heat exchangers with respect to the temperature, the flow and the pressure, the heat transfer coefficient (k) and the cv-number were tuned until the values agreed with "real" values. This was a rather cumbersome and time consuming. One reason for this was that the temperature equations easily turned unstable. The equations are only tuned for the production phase because it is the most interesting phase to simulate. This means of that in the other phases, the temperature may not agree fully with the correct temperature.

A solution to the problem of the equations turning unstable would be to include a set of equations for each phase. Each set of these equeations can be tuned for one particular phase and

they would agree with the correct temperatures in every phase. There are two reasons why we have not included this. First, the deviation from the correct temperatures is rather small and does not affect the simulation in any important way. Second, there would be an awful lot of equations to write.

### Steam injector

In the steam injector, steam is directly injected into the water line thus increasing the water temperature. We have assumed that the injected steam adds a small flow of water to the incoming water. The pressure in the steam injector is determined by the pressure of the injected steam. Also the temperature of the injected steam is determined by the pressure of the steam through the law

$$T = c * P$$

where

T = temperature of the steam
P = pressure of the steam
c = characteristic constant for a particular steam

In our case c = 0.01098 (water steam). This law is only true for a constant volume. In G2 we have implemented the following state equation to include a dynamic behavior in the steam injector:

$$T_{k+1} = 0.95T_k + 4.55P_k$$

This equation has a fast dynamic behavior which corresponds well to the fast increase in water temperature achieved when injecting steam into water. The temperature of the outgoing water is the mean value of the temperature of the steam and the temperature of the incoming water. This may not be physically correct but it gives a fairly realistic behaviour.

### Sources

The sources only have predefined values for the temperature, the pressure and the maximum flow of the liquid (or steam) which they provide the simulation model.

### Pipes and connections

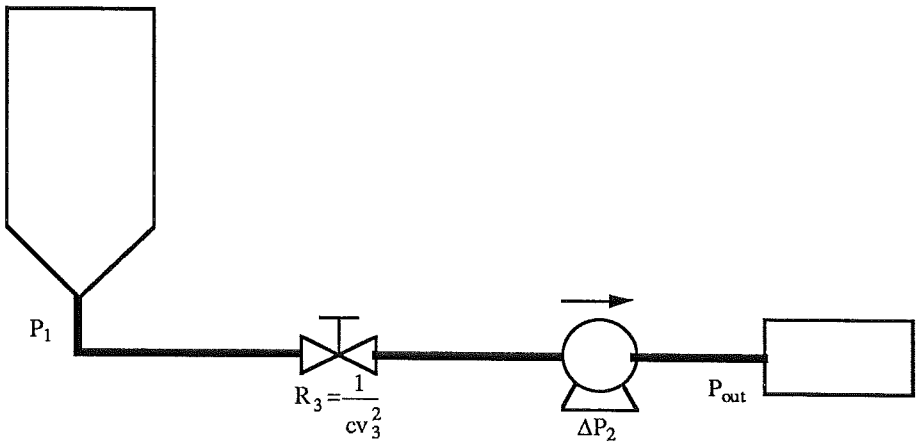
There are three kinds of pipes used in the model: product pipes, water pipes and steam pipes. All three are modelled as ideal pipes with no losses, i.e. no pressure drops , no decrease in the flow and no losses in the temperature.

## 5.3 How the simulation works

This section will explain how the simulation of flow, pressure and temperature in the model works.

### Flow and pressure simulation

To be able to simulate the flow and pressure in the process we use a method, similar to the method for calculating voltage and current in an electrical circuit. The different lines in the process, can be seen as lines where the pressure at both ends is known, with pumps that increase the pressure and with valves and heat exchangers that decrease the pressure. If for example you know the pressure at both ends and you also know the total flow resistance in the line, you can then calculate the flow in the line. The flow resistance in the line is the sum of the square of the inverse of all the cv-numbers in the line. The idea is best illustrated by an example:




Suppose the following values:

$P_1 = 2$

$\Delta P_2 = 1$

$P_{out} = 4$

$cv_3 = 0.5$

$\rho = 1$

$\Delta P_3 = P_{out} - P_1 - \Delta P_2$ $\Delta P_3$ is also equal to: $\Delta P_3 = \rho R_3 q^2$

This gives us the flow:

$$q = \sqrt{\frac{P_{out} - P_1 - \Delta P_2}{\rho R_3}} = 0.5$$

Because of the increase in flow due to the injected steam in the steam injector, the water line has to

be split into two separate lines, one that starts at the water tank and ends at the steam injector and one that starts at the outport of the steam injector and ends at the water tank.

For the same reason the product line also has to be split into two separate lines due to the packing machine. One line that starts at the product tank and ends at the packing machine and one that starts at the packing machine and ends at the product tank.

For each of these four sublines there exists a global function that calculates the total resistance and the flow. The flow-functions are called at the beginning of each line, for example in the product tank. This results in a call to the resistance-function that adds the flow resistances for the components in the line. Then the flow can be calculated by dividing the pressure difference between the ends with the total resistance. When the flow in a line is known (at the start point) the flow and pressure can propagate (in the same direction as the flow) through the line and the pressure at different points can be calculated. Each valve and heat exchanger can calculate their own pressure drop.

There is one exception from the rule that the flow and pressure always propagate in the same direction. This happens between the outport of the packing machine and the valve v74 (through valve v71) where the packing machine fetches its values of flow and pressure from the valve v74. Because v74 is a constant pressure valve it wants to regulate the pressure at the outport of the packing machine and therefore we can not have a predefined constant value at the outport of the packing machine. By tuning the valve v74, it is now possible to control the amount of product that is recycled from the packing machine back to the product tank.

In the following, we will describe the simulation of flow and pressure in the product and water line in more detail.

### Simulation in the water line

The water line in the process is shown in Fig. 5.2 on page 28. The water line, as mentioned before is mainly used to heat up the product to the sterilisation temperature 137 °C. It is also used to cool the product before it goes to the packing machine.

There is a valve v61 that directs the water flow to either the heat exchanger IV or the heat exchanger II depending on the actual phase of the process. Since the two heat exchangers have different flow coefficients, this affects the total flow resistance in the line. This means that the global function that calculates the resistance must know the current phase for the process, i.e the components involved in the "current" line.

The valve o2 controls the flow in the first part of the water line, from the water tank to the steam injector. The flow in the second water line, from the steam injector back to the water tank is controlled by the valve v65. The flow back to the water tank is always greater than the flow into the steam injector. This is because steam is injected directly into the water. The flow of the injected steam is given by the difference between the reference flow of v65 and the actual flow of o2 and

not by the current flow difference between the two valves. The reason why we use the reference value of v65 instead of the actual flow is that when using the current flow, the calculated flow from the flow functions turned unstable. We do not have a complete answer as to why this happens. We believe the reason is the following. When the two valves have calculated their new valve positions a new steam flow is calculated. This will generate a new pressure of the steam into the steam injector. The flow functions use this new pressure when calculating a new flow. The valves o2 and v65 are now calculating a new valve position to correct the new, possibly wrong flow and this will generate a new steam flow and in turn a new steam pressure, etc.

The valve v44 controls the steam pressure into the steam injector. It is controlled by a PID-regulator that measures the temperature in the holding cell. By regulating the pressure of the steam into the steam injector, the temperature in the outgoing water is controlled and this in turn controls the temperature in the holding cell (through the heat exchanger I).

### Simulation in the product line

The product line is shown in the Figure 5.3 on page 29. In the production phase, product is circulating in this line. In the other phases sterile water or different kinds of detergent liquids are circulated.

In the production phase, the product from the product tank flows through the heat exchanger III for preheating and then to heat exchanger I for sterilization at 137 °C. Then the liquid flows through the heat exchangers II, III and VI to be cooled off before it goes to the packing machine. There is also a line from the packing machine back to the production tank. When in production, there must be a small over pressure in the packing machine. This will result in some product that is recycled back to the product tank.

In the first production line the valve v78 regulates the flow. This valve is a pressure regulating valve and it tries to keep a constant pressure between v78 and heat exchanger VI by regulating its valve position. Valve v78 will thus determine the flow. There are some phases when v78 is inactive (see the activation chart in Appendix A). In the second production line there is a valve v74 that tries to keep a predefined pressure in the packing machine. This will determine the flow in this line. Valve v74 is of the same type as v78. V74 is only active in the production phase. Except in the production phase these two valves are never active simultaneously.

In all phases except the production phase, the packing machine is inactive (even though it is connected) and acts just like a pipe. This is the reason why the valves v74 and v78 never are active at the same time, except in the production line.

Depending on which valve of v74 and v78 that is active, the position of the valve v20, and the position of some other valves (near the production tank), the total flow resistance in the two lines will be very different. The global functions that calculate the flow must take care of each of these different cases to be able to calculate the right flow.

### Temperature simulation

Most of the temperature simulations are done in the heat exchangers (see Section 5.2 about the heat exchangers). In both the product and water tank the temperature of the various inflows into the tanks are mixed (see Section 5.2 about the tanks). This temperature propagates in the different lines in the same direction as the flow.

### Burn-on simulation

The burn-on is only simulated in heat exchanger I, because the effect of the burn-on is most observable there. Of course the burn-on simulation is only active in the production phase. The burn-on is simulated in two different ways. First, through manipulating the heat transfer coefficient of the heat exchanger and second through manipulating the cv-number.

The heat transfer coefficient is manipulated through subtracting a state variable, temp-burn-on-state-variable, from the coefficient. The state variable increases with time. The increase rate is given by another variable, temp-burn-on-characteristic. The value of this variable depends on the actual product in the process. For a product that causes fast burn-on, this characteristic variable has a greater value than for a product that causes slower burn-on. Decreasing the heat transfer coefficient will require a higher temperature on the heating side of the heat exchanger to keep the temperature on the other side of the heat exchanger constant.

The cv-number of the heat exchanger is manipulated in the same way as the heat transfer coefficient (except that the variables now are called pressure-burn-on-state-variable and pressure-burn-on-characteristic). Decreasing the cv-number is the same as if the cross area of the pipe in the heat exchanger gets smaller, which is what actually occurs during burn-on. This will result in an increase in the pressure drop.

As mentioned before, the burn-on is only simulated in the production phase; but, in all phases after the production phase the effect of the burn-on remains. The effect is still there until the process has been cleaned. There are two different kinds of cleaning in the Steritherm process, an intermediate cleaning (AIC) and a final cleaning (CIP). After the intermediate cleaning the effect of the burn-on is reduced by 50 %. This is managed in G2 by using a whenever rule that tests when the current phase is intermediate cleaning. After a final cleaning the whole effect of the burn-on has vanished.

## 5.4 Conclusions

It has been rather difficult to get a reasonable and well suited structure among the simulation equations in the simulation model. On the other hand, the construction of the different object classes and their subclasses has been made rather structured. The possibilities in G2 to use generical simulation formulas for the different objects used in the model is a nice way to write simulation formulas. One can easily create new objects and connect them to each other without having to write new simulation formulas for each of the new objects. One can also delete objects without rewriting any simulation formulas. However the generic simulation formulas have been rather difficult to use in the simulation model of the Steritherm process. The reason for this is to some extent restrictions in the G2 simulator, but the main reason is the complexity of the Steritherm process. Every object of a specific class is connected to several other types of objects. The problem here is that all the classes have different names for the same physical magnitudes. Take for example a valve with one inlet and one outlet. It has among others the attribute inflow and outflow. Compare this with a valve with two inlets and two outlets. This valve has the corresponding attributes inflow1, inflow2, outflow1 and outflow2. Due to this problem it is not possible to inherit some important attributes from classes on a higher level in the hierarchy.

Due to the difficulties mentioned above, the generic simulation formulas would have to state all possible ways to connect the different objects. This would result in very huge, clumsy and unreadable generic formulas which would probably not be general either, which is the point of generic formulas.

Because of this, all simulation formulas have been placed in the respective objects' attributes with the exception of the formulas for the heat transfer in the heat exchangers and the formulas for the PID-regulator; which are generic. The drawback of not using generic formulas is that it is more difficult to connect new objects in a model because new simulation formulas have to be written for all attributes of that object. Problems also arise when objects are deleted from the model.

We can not do anything about these problems until the simulator in G2 is changed. However, the most important aspect of the simulation model is that it provides quantitatively correct data to the control system and not that the simulation equations are well structured.

# 6 The control system

The control system is the most important and interesting part of this project. It contains the control and supervision of the process. In addition, it contains a lot of information and knowledge about the process. The control system is constructed to allow qualitative heuristic knowledge and underlying functional knowledge to be represented.

The construction of the control system is based on the common knowledge-base concept. This means an integration of knowledge-based techniques and conventional control systems. This common knowledge database consists among other things of process components, control system components , functional and structural descriptions , alarm and fault trees, control sequences, etc.

The knowledge description in the common database is made in a hierarchical and object-oriented manner. Multiple representations (views) are used to handle the different types of knowledge in the system. Production rules are used to represent heuristic, qualitative knowledge.

Section 6.1 describes the hierarchical levels and the different kinds of views used in the system. Section 6.2 gives an overview of the process schematic seen by the control system. In Section 6.3, a description of the control sequences used in the Steritherm process is given. In Section 6.4, the regulators used in the process are described. Section 6.5 describes the alarm analysis that is implemented in the control system. Finally, Section 6.6 describes the supervision of the burn-on.

## 6.1 Hierarchical levels and views

In this section, the different hierarchical levels of the process and the views at each level are described. First, we give an overview of the concept of hierarchical levels and views. Second, we describe the hierarcical levels and views that have been implemented in the project. Last, we describe the implementation.

### The concept of hierarchical levels and views

Hierarchical levels are used to represent the Steritherm process at different degrees of resolution. The highest level in the hierarchy represents the whole plant, which in this case is the dairy. The lowest level represents the basic entities, e.g. pump-rotator, screws, plates in heat exchangers, etc. With hierarchical levels, knowledge and other kinds of information can be stored at the most appropriate level. The hierarchical levels lead to a tree structure where the root corresponds to the whole plant and the leaves correspond to the basic components. In this structure, geographically and logically related components can be grouped together.

This simple structure is not enough. Several views of the process are needed to represent all

information of importance.

An object, e.g. the Steritherm process or a pump, contains knowledge and information of different kind that belongs to the object. This knowlege or information can be the inner structure, a functional description or a photo or textual information of the object. These different kinds of knowledge about a single object are called multiple representations or views.

### The object-oriented model of Steritherm

At the highest level in the hierarchy there is a single object representing the whole dairy. On top of this object there is also a button named "views". When clicking on this button a graphical menu appears on the screen. When clicking on the object, but not on the button, the attribute table of the object appears. Figure 6.1 shows the top level view with the dairy and its menu.

The menu contains menu choices for the different views of the dairy object. These menu choices are represented as icons. On top of each icon is a button. When clicking on this button, the view that the icon represents pops up. When clicking on the icon but not on the button an attribute table of the icon appears. This table contains general information that concerns this particular view.

The attribute table of the object consists of various information about the object. The table contains references to the views of the object. It may also contain more information that characterizes the object. For example, the attribute table of the object that represents the Steritherm

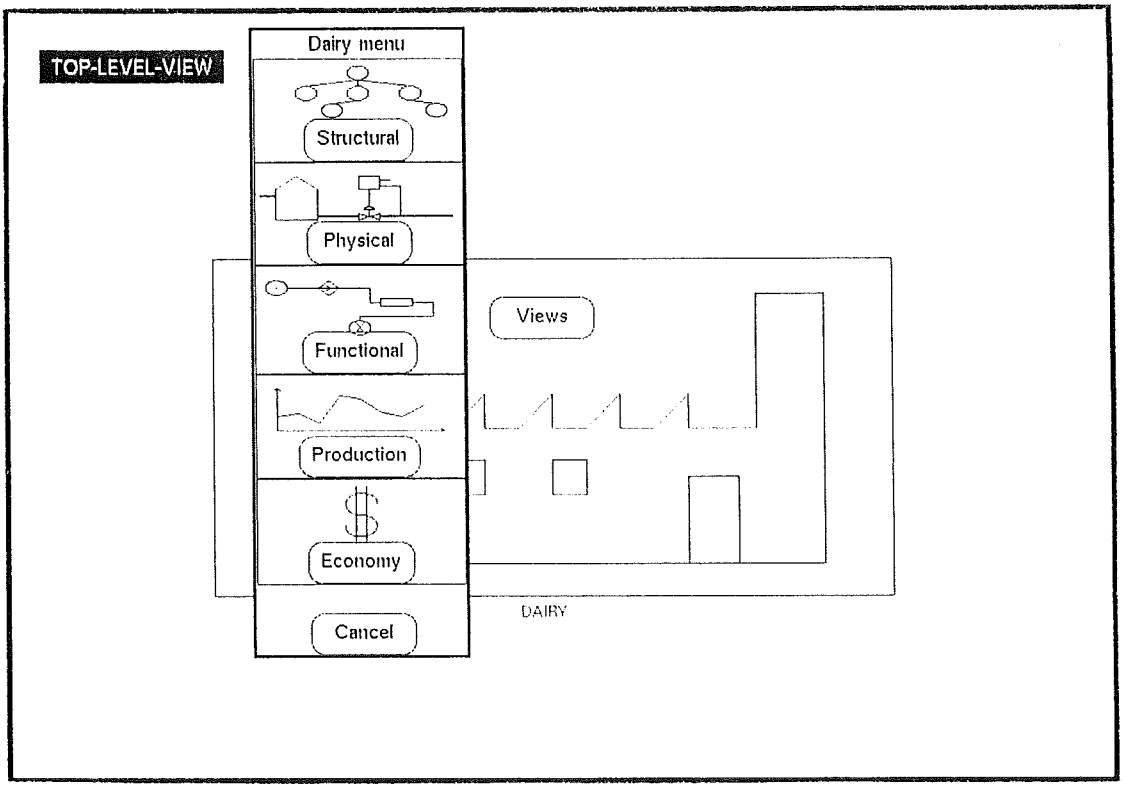



**Figure 6.1** The top level view showing the dairy object and its menu

process consists of in addition to the view references, the time intervals that determine how long the process shall remain in the different phases that the process runs in.

On every view (except the top level view) there are two buttons named "Superior view" and "Superior menu". With these button you can either go to the previous view in the hierarchy or to a view that is on the same hierarchical level as the current view (via the menu on the superior view).

In the rest of this section, we will explain the hierarchical levels and the views at each level which have been implemented in the control system.

The menu of the top level object (the dairy) consists of the following menu choices (see Figure 6.1).

* **Structural**: This view gives a more detailed description of the inner structure of the dairy.

* **Production**: This view may show information concerning raw product consumption, production of the different end products, process or line utilization and status, historical statistics, production plans, etc.

* **Economy**: This view may show the economical aspects of the dairy, e.g. return on investment, profit, down-times, consumption of electricity, steam, water, etc.

* **Physical**: This view may show photos of the plant, room and floor layouts, etc.

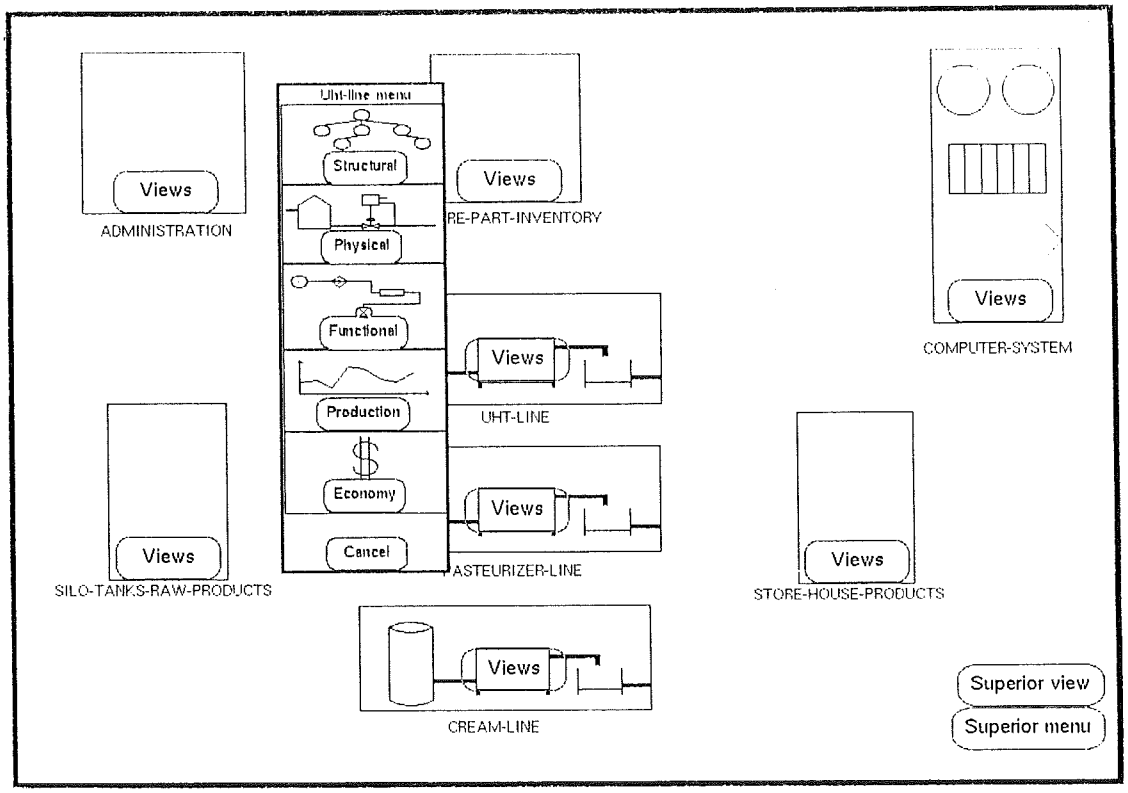



**Figure 6.2** The structural view of the dairy object

* **Functional**: This view gives, perhaps in textual form, a functional description of the plant.

Of these views, only the structural view is fully implemented in the system. The other views are only present to show the possibilities. The structural view consists of a number of different objects, that give a more detailed description of the dairy. Figure 6.2 shows the structural view of the dairy and the menu of the UHT-line object. The objects at this level are briefly described below.

* **Administration**: This object consists of information regarding the staff of the dairy, e.g. employments, shifts, names, adresses, wages, etc.

* **Silo tanks**: In this object, there is information about the different raw products used in the dairy.

* **Spare part inventory**: This object represents a database about all spare parts that are available in the dairy.

* **UHT line**: This object represents the production line for UHT treated products.

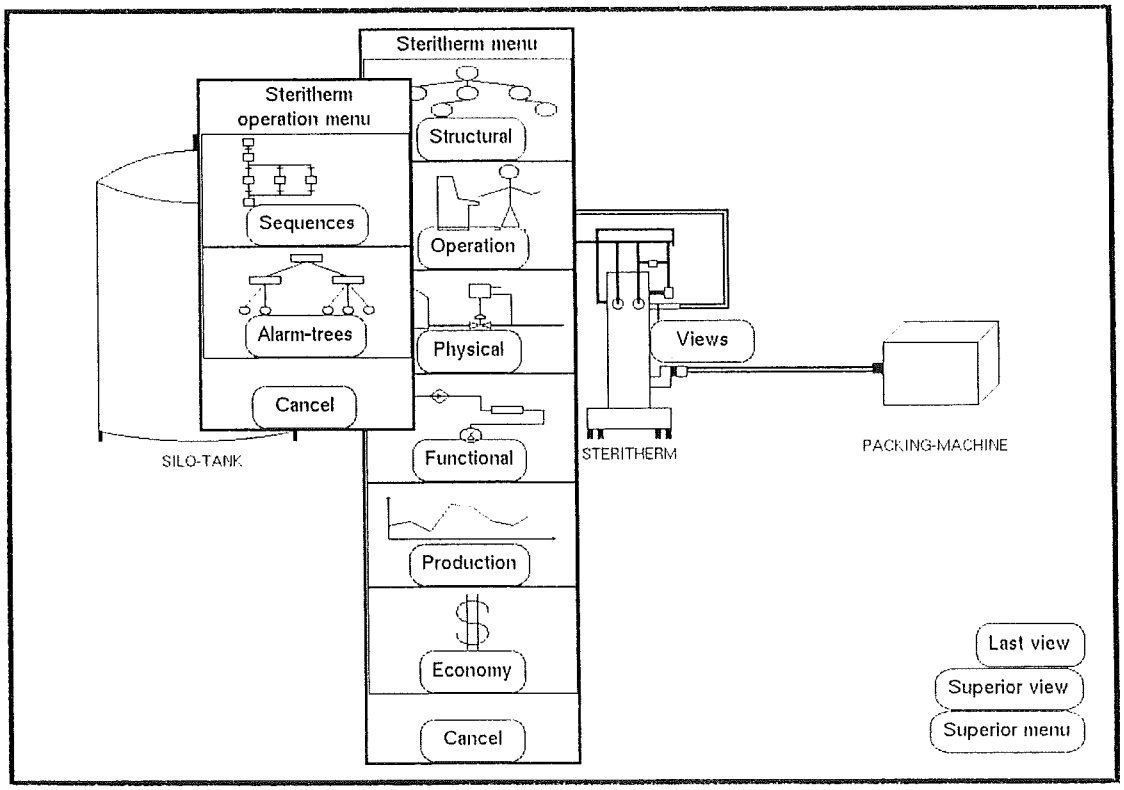



**Figure 6.3** The structural view of the UHT-line object

* **Pasteurizer line** and **cream line**: These two objects represents two other kinds of production lines in the dairy.

* **Computer system**: This object contains information about the whole computer system used in the dairy, e.g. networks, terminals, process control units, operator consols, etc.

* **Store house products**: This object consists of information about the end products.

Of these objects, it is only the UHT-line that has further internal structure. The menu of the UHT-line contains the same type of views as the dairy menu. Of these menu choices it is only the structural view that really represents a view. The view is shown in Figure 6.3. This figure also shows the menu of the Steritherm object and the submenu for the operation choice. There are three objects in the structural view of the UHT-line

* **Silo tank**: This object contains information about the raw products used in the UHT-line.

* **Steritherm process**: This object contains information about the Steritherm process.

* **Packing machine**: This object consists of information about the packing machine.

Again it is only the Steritherm object that has a further structure and a complete menu. This menu contains an additional menu choice called operation. This menu choice in turn contains a submenu with two alternatives (see Figure 6.3). There are four of these menu chioces that are implemented.

* **Structural**: This view describes the process schematic. The schematic contains all process components (e.g. pumps, heat exchangers, valves, tanks, etc.) and control system components and how they are physically connected. Also in this view the process components have further structure and additional views. The process schematic and its views are described in Section 6.2.

* **Sequences**: This view contains the sequence net that controls the operation of Steritherm. The sequence net is described in Section 6.4.

* **Alarm trees**: This view consists of the fault objects and rules that handle the fault analysis, i.e. tries to locate the cause of an error that has appeared in the process. This view is described in Section 6.5.

* **Production**: This view consists of diagrams that presents the time the Steritherm process

has been in the production phase compared to all other phases (to the AIC phase or the CIP phase). This gives an estimation of the production efficiency of the process.

### The implementation

The class definitions for all hierarchical objects, e.g. dairy object, UHT-line object, etc., are found on the top level workspace view-object-definitions in G2. These classes are hierarchically organized. The dairy-class is at the highest level. This class defines the dairy object. Subclasses to this class are classes describing objects that are found on the structural view of the dairy object. Examples of classes at this level are UHT-line-class, Administration-class, Computer-system-class, etc. Subclasses to UHT-line-class are among others, the Steritherm-process.

The class definitions for the (icon) objects that makes up the menus, containing choices for the various views of an object, are found on the top level workspace called view-menu-class-definitions. The superior class to all menu classes is called workspace-objects and the classes that defines the structural view, the physical view, etc. are subclasses to this class.

The subworkspace of each hierarchical object (i.e. dairy, Steritherm, etc.) contains the menu that consists of the views for the object. This menu appears when you click on the button that is positioned within the bounds of the object. The menu consists of a number of icons (objects) that

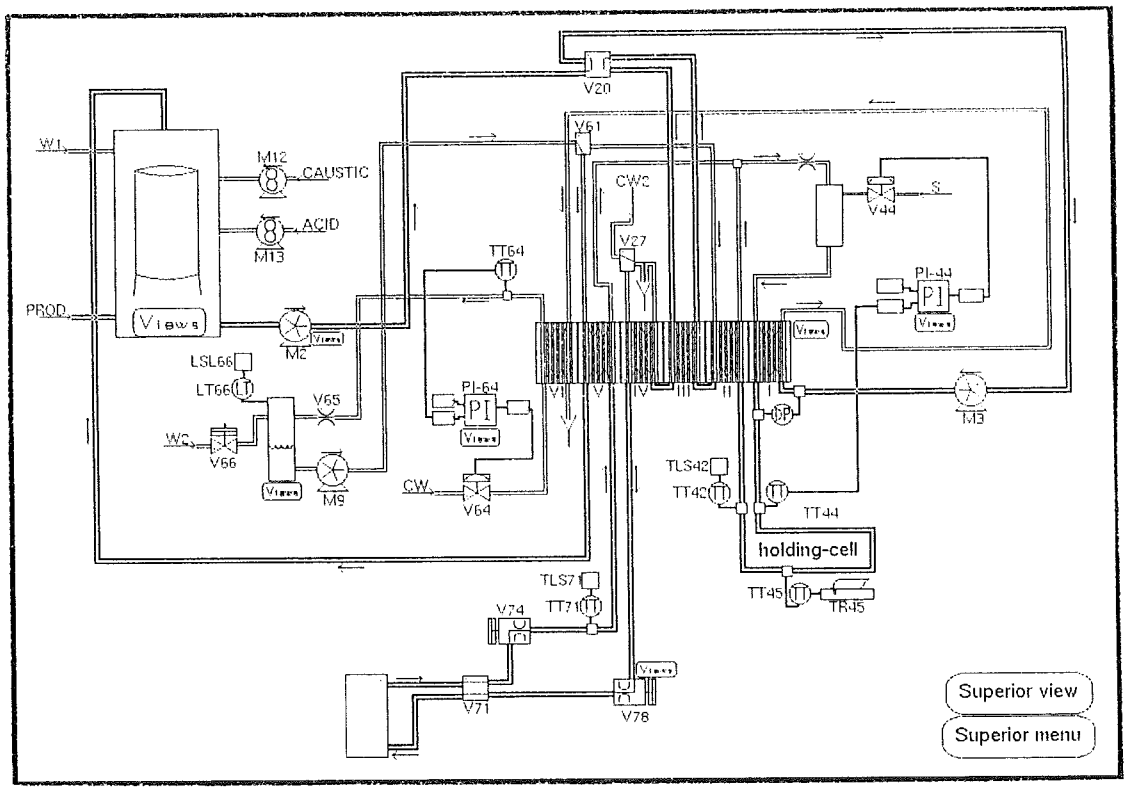



**Figure 6.4** The process schematic, i.e. the structural view of the steritherm object

each represents a specific view. There is also a button placed on each of these icons. The views of a hierarchical object are implemented in the subworkspace of the (icon) objects that makes up the menu. The views appear when you click on the buttons positioned on each choice (icon) in the menu

The highest level of the hierarchical structure implemented in the system, i.e. the dairy object, is found on the top level workspace called top-level-view.

## 6.2 The process schematic

This section discusses the inner structure of the Steritherm process seen from the point of view of the control system. The process schematic, shown in Figure 6.4, is a graphical representation of the process. It contains all process components such as pumps, heat exchangers, valves, etc. and their connections. It also contains PID regulators, alarm limit blocks, etc. The graphical representation is the structural view of the Steritherm object (see Section 6.1 about the the object-oriented model of Steritherm).

### Objects and classes

This process schematic looks very much the same as the schematic of the simulation model (compare Figure 5.1 which represents the simulation schematic to Figure 6.4). There are of course no simulation formulas "behind" the objects in this schematic (except in the case of the regulators, see Section 6.3) and the object attributes contain a completely different kind of information. This information is mainly of a qualitative nature. The objects also contain hierarchical levels with different views (see next section). In addition to the usual process components, there are also two regulators, alarm sensors and a differential pressure transmitter. The regulators that control the valves v44 and v64 are described in detail in Section 6.3. The alarm sensors are used to indicate an alarm to the control system, e.g. when the temperature in the holding cell gets to low. The alarm sensors are connected in the holding cell, after the packing machine, and at both tanks (TLS42, TLS71, LSL66 and LSL65). The differential pressure transmitter (DP) is connected at the production side of heat exchanger I. This transmitter is used by the control system for supervision of the burn-on (described in Section 6.6). There are no indicators in this schematic (as in the simulation schematic). The reason for this is that they are not connected to the control system. However, the sensors that really are connected to the control system are present in the schematic. Since the real Steritherm process is not available, these sensors collect their values from the simulation model.

The classes that define the objects on the schematic are found on the top level workspace called schematic-object-definitions. The classes are built up in a hierarchical manner similar to the

classes of the simulation model. At the highest level there is a class called view-object. Under this class there are process-equipment, production-tank, and a number of sensor classes. Under process-equipment, classes for all process components, alarm sensors, and the regulators are defined. The sensor classes define the different sensors used in the process, e.g. temperature transmitter, level transmitter, differential pressure transmitter, etc. The class named production-tank defines the object at the upper left corner in Figure 6.4. This object is used to replace the production tank with its surrounding valves. With this object, one will have a clearer view of the process. In the next section about the hierarchical levels and views, this object will be described in more detail.

All classes (except the sensors) have attributes of the same kind. The attributes refer to the views that a specific object can have (see next section for more details). All classes under the class process-equipment have in addition an attribute called status. Status can have one of the two values "on" or "off". Status indicates e.g. if a pump is running or not, one of two directions of a valve or if a regulator valve is active or not. Status receives its value from the sequence net in the control system. The value of each status attribute depends on the current phase of the process. When status receives a value, this value must be passed on to either the simulation model or to the real process if it is present. This is done with "whenever-rules". These rules fire when status receives a new value and put the same value into the simulation model. It is easy to change these rules to direct the value of status to the real process instead. These "whenever- rules" are defined in the top level

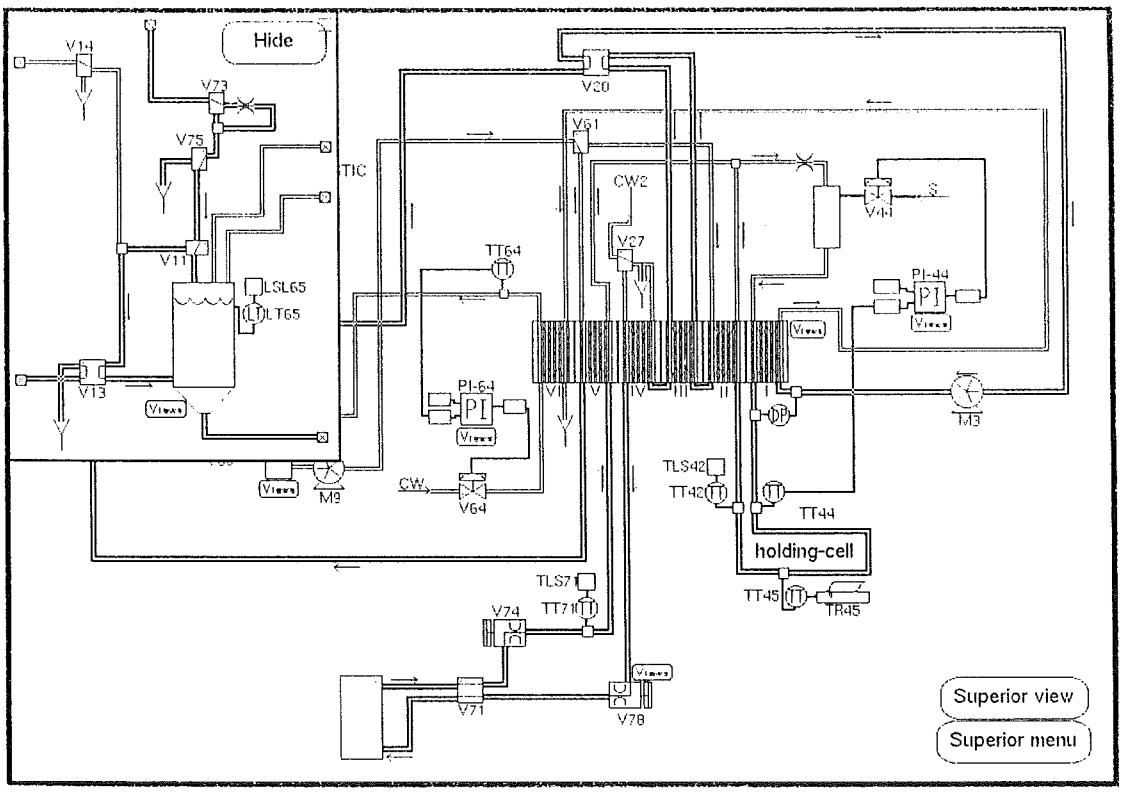



**Figure 6.5** The process schematic with the product tank and its surrounding valves

workspace called animation-and-set-rules. Except for the regulators, the only connection between the control system and the simulator/real process is done through these rules and through the transmitters mentioned before.

### Hierarchical levels and views

As mentioned before, the process schematic (Figure 6.4) represents the structural view of the Steritherm object (see Section 6.1 about the object-oriented model of Steritherm). The objects on this view have in turn hierarchical levels and views. The structural view of the object that represents the production tank with its surrounding valves is shown in Figure 6.5. This view shows in detail the production tank, its surrounding valves and how they are connected. Also these objects like the other objects on the schematic have hierarchical levels and views.

Common views of the objects are the following.

* **Structural**: This view contains the inner structure of the object, e.g. a pump consists of a rotor, a stator, a pump wheel, bearings, etc.

* **Physical**: This view contains photos of the object.

* **Functional**: This view consists of a functional description of the object.

However, other views are possible. The regulators for example, have the following additional views.

* **Trends**: This view contains graphs of the set-point, measured value, and control signal of the controller.

* **Implementation**: This view consists of the G2 code that implements the regulators.

* **Parameters**: In this view, it is possible to change the value of the parameters of a regulator.

To make the menus of the objects appear on the screen, one must click on the button marked "Views" located nearest the object. Clicking on this object will make the attribute table appear. As with the attribute tables for the other objects with hierarchical levels, this table also contains references (i.e. the names of the icons in the menu) to the views of a object. The buttons used here are not the usual G2 buttons used before. The reason for not using the G2 buttons, is that they are far too big compared to the objects in the schematic. The buttons used are ordinary objects, which

can be decreased to a reasonable size. This means that the menu of an object must be located in the subworkspace of the button and not, as before, placed in the subworkspace of an object. The reason for this is, when clicking on an object, it is only possible to make its own subworkspace appear. G2 does not allow other subworkspaces to appear when clicking on objects. This is only possible with buttons.

### The animation

To get a survey of the status of the different components in the process (i.e. if the pump is running, the position of a valve or the level of a tank), graphical animation objects are used. For the pumps, an animation object that looks likes a rotator, is positioned in the middle. When the pump is running the rotator rotates, otherwise it is still. For the valves there are different animation objects that indicate the different positions of the valve. For the tanks, an animation object that look likes a wave, is used to indicate the levels (see Figure 6.5). The classes that define the animation object are found on the top level workspace called animation-object- definitions.

Rules are used to make the animation work properly. For the valves, "whenever- rules" are used. These rules fire when the status attribute in a valve receives a new value and they move the proper animation object to the right position at the valve. For the pumps, scanned "if-rules" are used. If the status attribute of a pump has the value "on", the rotator will be rotated every second (that is whenever the rule is scanned). Also for the tanks, scanned "if-rules" are used. In this case the status attribute can not be used, instead the level transmitters are used to get the actual level of a tank. The wave object changes position every second (i.e. when the rules are scanned). The rules that handle the animation are found on the top level workspace called animation-and-set-rules. It should be mentioned that animation is also used in the simulation schematic. This animation works in the same way and it is handled by the same rules as the animation described in this section.

## 6.3 The regulators

In the Steritherm process there are two PI-regulators used to regulate the temperature in the holding cell and in the water tank. Actually they are PID-regulators but the derivation parts are inactive. The regulators are placed in the process schematic of the control system (the structural view of the Steritherm object).

### Function

The first regulator PI-44 is used to control the temperature in the holding cell. This is done through

valve v44, which regulates the pressure and with that the temperature of the steam into the steam injector. By regulating the temperature in the water line like this, the temperature in the holding cell can be regulated through the heat transfer in heat exchanger I. PI-44 receives its input values from transmitter TT44 connected at the holding cell.

The second regulator PI-64 is used to regulate the temperature of the recirculated water into the water tank. This is managed by regulating the valve v64, which will regulate the flow of cold water through heat exchanger V and with that the temperature of the recirculated water. PI-64 receives its input values from transmitter TT64.

The equations that implement the regulators are generic (Figure 6.6 shows the code in G2) and placed in the subworkspace of the object definition called PID (on the workspace called schematic-object-definitions).

The regulators receive their input values from interface blocks, which in turn receive the desired values from the simulation model. The control signal that the regulator calculates is also sent to an interface block that sends it to the simulation model. Thanks to the interfaces it has been possible to write the equations for the regulators such that they are generic and independent of their associated application.

Both regulators have their parameters tuned for the production phase and they regulate nicely in this phase. In other phases, the parameters might not be tuned as well as in the prodution phase and this may result in small oscillations in the control signals. This corresponds to the situation in a real Steritherm process.

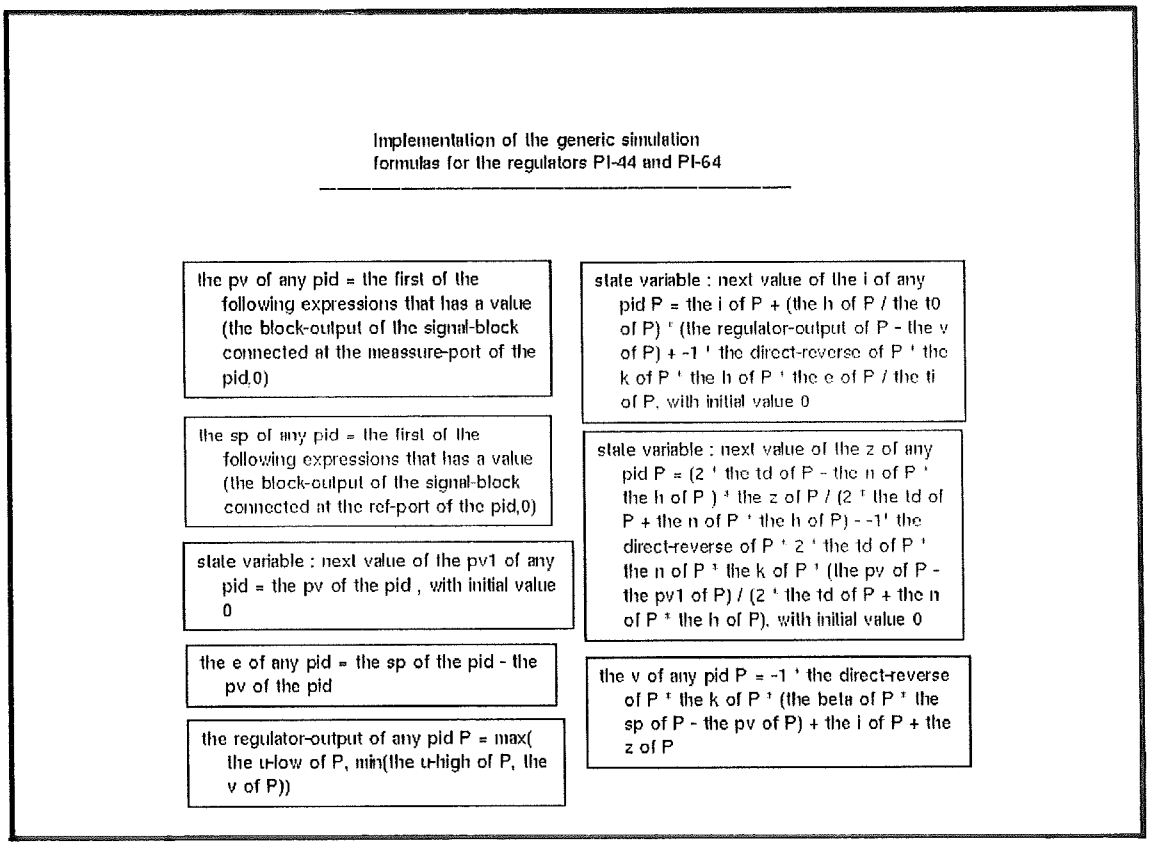



**Figure 6.6** The equations in G2 that implements the regulators

### Implementation

The classes for the regulators are found on the top level workspace schematic-object-definitions. The class regulator-blocks is a subclass of the class process-equipment. There are no attributes in this class but it has the two subclasses signal-block and PID. The class signal-block is an interface block between the regulator and the simulation model.

PID is a regulator class that describes a discrete PID regulator. The simulation equations for this class are generic and are found in its subworkspace. The class has the following attributes: Sp, pv, pv1, z, i, e, k, ti, td, n, t0, h, beta, direct-reverse, u-low, u-high, v, regulator-output and finally some attributes that are references to the views of the regulator.

Sp is the reference value of the regulator. Pv and Pv1 are the current and last valves of the measured process variable. Z corresponds to the derivative part and i corresponds to the integral part of the regulator. E is the regulator error. K is the regulator gain. Ti and td are the integration time and the the derivation time respectively. H is the sampling time. Direct-reverse indicates direct or reverse feedback. U-low and u-high are the minimum and the maximum of the control signal.V is the calculated output. Beta, n, and t0 are additional parameters further described in Åström (1987).

## 6.4 The sequence net

The sequence net in the control system contains a number of sequences for handling the different phases used in the Steritherm process and also the transitions between the different phases. The sequences are implemented in a Grafcet-like style (Lannefors, 1987). The sequence net that controls the operation of the Steritherm process is found in the operation view of the Steritherm object. This view is shown in Figure 6.7. On top of the figure there is a simple control panel for the operator.

The process is started by clicking on the button called "Start up Steritherm" on the operator's control. There is a marker that always indicates the current phase of the process. In Figure 6.7 the process is in the production phase.

### Function

The sequence net is constructed with steps and transitions according to the Grafcet formalism. In the steps, operations on the process are performed (e.g. a valve gets a new position) and in the transitions there are conditons for getting from one step to another (e.g. when the tank is empty, go to step empty-tank). In principle, there are two different kinds of objects describing a step. These are structural step and terminal step. A structural step consists of a subsequence that in turn can

consists of steps, transitions and possibly subsequences. In Figure 6.8 the subsequence of the sterilizing step is shown. Also a subsequence to the heating-sterilizing step in the subsequence of the step sterilizing is shown. The subsequence of a structural step is placed in the subworkspace of the same step.

It is only in the other type of steps, called terminal step, that operations on the process really are performed. This is done with "initially-rules" placed in the subworkspace of the step. These rules make a "conclude" action that sets the status attributes of the process components (objects) that are found on the process schematic. An example of a "initially-rule" in a terminal step is shown below.

```
initially in order
    conclude that m2 is on and
    conclude that m3 is on and
    conclude that m9 is on and
    conclude that m12 is off and
    conclude that m13 is off and
    conclude that v11 is on and
    conclude that v13 is off and
    conclude that v14 is off and
    conclude that v20 is on and
    conclude that v27 is on and
    conclude that v44 is on and
    conclude that v61 is on and
    conclude that v64 is on and
    conclude that v66 is on and
    conclude that v71 is on and
    conclude that v73 is off and
    conclude that v74 is off and
    conclude that v75 is on and
    conclude that v78 is on and
    conclude that mode = 5
```

(1)

Transition objects contain the transition conditions. These conditions are found in the subworkspace of the transition objects. There are two different kinds of transition rules. Both transition conditions are made up of "when-rules" that are scanned every second. Below, examples of these two kind of rule, are given.

```
when elapsed-time(p1-prod-in,T4-5) then
    conclude that the transit of p1-transition2
    is true
```

(2)

```
when LT65 <= min-level then conclude that
    the transit of p1-transition1 is true
```

(3)

The first rule, consists of a time condition that must be fulfilled before a transition can take place. The second rule consists of a condition in the process that must be fulfilled, e.g. when the level of

a tank is less than 0.1. The time condition uses a boolean function which returns "true" when the specified time has elapsed. The function takes two arguments and looks like

elapsed-time(<previous-step>,<time>) (4)

The first argument, <previous-step> indicates the name of the step where the process shall remain for the time that is specified by the second argument, <time>.

When the transition condition is fulfilled, the subworkspace of the step connected just before the transition is deactivated. Also, when the transition is fulfilled, the subworkspace of the transition becomes inactive along with other transitions that may be connected to the same step. Deactivating the subworkspace of a step or transition means that the rules in the subworkspace become inactive. After this is done, the subworkspace of the step connected directly after the transition becomes activated and the marker is moved to this step. Finally, the subworkspaces of each new transition that is connected after this new step, become activated. Activating the subworkspace of the step (if terminal) will make the rules in this subworkspace activated. In the subworkspace of a terminal step, there may be both "initially-rules" that fire directly when the subworkspace becomes activated and usual scanned "if-rules". Activating the subworkspace of a transition will make the rule in the subworkspace of each transition check the transition condition. When this is true, the above procedure is repeated again.

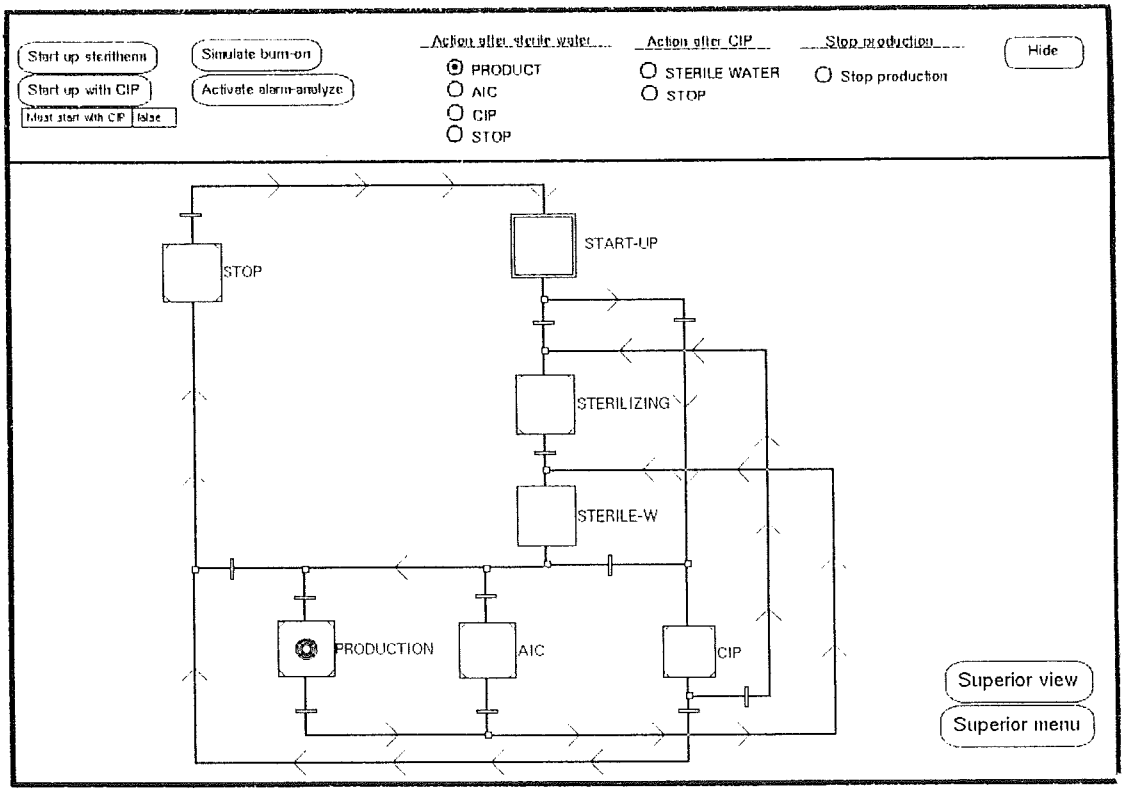



**Figure 6.7** The sequence net and the operators control

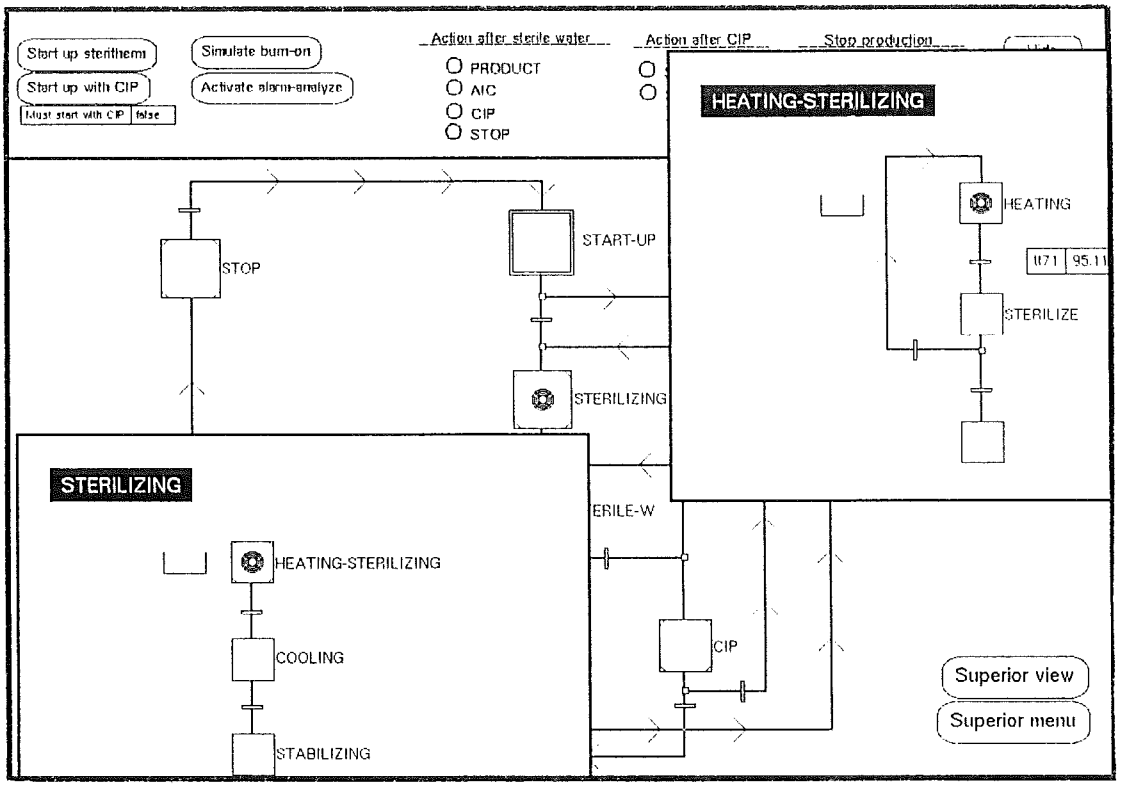



**Figure 6.8** The subsequence of the sterilizing step and the subsequence of the heating-sterilizing step

How it works, when the subworkspaces of steps and transitions are deactivated and activated and how the marker is moved, will be explained in detail in this section about the Grafcet implementation.

## Operator's control

In the system, there is a very simple operator's control. With this control the user can start the process, simulate burn-on and activate the alarm analysis. The operator's control is found at the top of Fig. 6.7. This control is only designed to make it possible to run the process under the development of the control system. The operator's control must not be confused with the real operator's interface. The button marked "Start up Steritherm" and "Start up with CIP" are used to either start the process from the beginning or to restart the process after a fault has occurred. In the latter case the process starts with a CIP (cleaning in place).

There is also a button for simulating the burn-on. When clicking on this button, a workspace pops up on the screen. This workspace contains different choices concerning the simulation of the burn-on. The user can choose to simulate the burn-on with or without noise and there is also the possibilty to interrupt the burn-on simulation.

The button marked "Activate alarm analysis" is used to activate the part of the control system

that handles the alarm analysis. This part tries to locate the cause of an alarm.

In addition, there are a number of buttons divided into three groups. These buttons are used to control the process. In the first group called "action after sterile water", the user can choose to run the process in one of the following four modes: production, AIC, CIP or stop. In the second group called "action after cip", the user can choose to enter either the sterile water mode or the stop mode. In the third group called "stop production" the user can stop the production phase when it is time for a cleaning.

### Grafcet implementation

The Grafcet implementation consists of objects that represent steps, transitions, and parallel transitions. The objects are graphically interconnected. The objects must have subworkspaces that can be activated and deactivated. In addition, there is an object that represents the marker indicating the current phase that is active. The administration of activation and deactivation of subworkspaces and moving the marker are handled by a number of generic rules in G2. In the subworkspace of a step there are either subsequences or rules that performe the desired operation on the process. In the subworkspace of a transition there is a rule with a transition condition that must be fulfilled.

### The Grafcet objects

The classes that define the objects that belong to the sequence net are found on the top level workspace called grafcet. On the highest level there are three classes called sequence-object, container and sequence-flow- line. These will be explained below.

#### Sequence-object

This class has no attributes but it has three subclasses: step, transition and marker.

Step has the following three attributes: time-of-enter, type, and empty-container. Time-of-enter receives the value of the current time when the subworkspace of a step becomes activated. This attribute is used to keep a step activated (i.e. its subworkspace) under a specified amount of time. Type is used to distinguish between two different types of step, namely structural and terminal. Structural means that the step contains a subsequence in its subworkspace while terminal means that the step contains rules (that perform operations on the process) in its sub- workspace. Empty-container states the name of the container object (explained below) that is found on the subworkspace of the step. This attribute only makes sense if the step is structural. This attribute and the container object are necessary to make the sequence net work properly (see this section about the generic rules for the sequence net). The classes that define the step and the transition have the value "activatable-subworkspace" in the slot called capabilities-and-restrictions. This is

necessary to make it possible for G2 to "forget" the rules on a subworkspace when it is deactivated and "remember" them again when it is activated. All "initially-rules" on activatable subworkspaces will fire every time the subworkspace is activated. The class step has also two subclasses called starting-step and structural-step. These only differs in the icons of the objects.

The class transition has only one attribute called transit. Transit is a boolean attribute that states if the transition condition in the subworkspace of the transition is fulfilled or not. This means that the rule in the subworkspace must set the transit attribute to "true" when the transition condition in the rule is fulfilled. See rules 2 and 3 above for two examples of transition rules. The class parallell-bars are used to split up a sequence in many parallel sequences. This kind of transition is not used in the sequence net in the control system and it will not be further explained.

The class marker has no attributes. The object of this class is only used to indicate which step is currently is activated.

### Container

An object of this class must be positioned nearest the first step in a subsequence. It is also used to hold the marker object when the subsequence is inactive. There are two attributes in this class, super-step and last-step. Super-step states the name of the possible structural step that contains the container in its subworkspace. The attribute last-step states the name of the last step in a subsequence.

### Sequence-flow-line

This connection class defines the connections used to connect the different kind of sequence objects into a sequence net.

## Generic rules for the sequence net

The generic rules that manage the sequence net are found on the top level workspace called grafcet. On this workspace there are a total of 12 rules. The rules can be divided in two categories. Rules in the first category are activated when the attribute transit in a transition object receives the value "true". The second category is invoked whenever a marker object is moved.

When a transition condition in a transition rule is fulfilled (and the attribute transit is "true") the following actions are taken by the generic rules.

1. The marker object is moved from the current step to the step positioned directly after the transition that just has been fulfilled.

2. The subworkspace of the "old" step is deactivated.

3. The subworkspace of the "new" step is activated.

A rule that performs the actions mentioned above is shown below.

```
whenever the transit of any transition t
   receives a value
and when the type of the step st
   connected at the previous-step of t is
   terminal
then move the marker nearest to st to (the
   icon-x-position of the step connected at
   the next-step of t, the icon-y-position
   of the step connected at the next-step
   of t)
and deactivate the subworkspace of st
and activate the subworkspace of the step
   connected at the next-step of t
```

(5)

If the "old" step is of the type structural, additionaly actions must be performed by the rules. The subworkspace of the last step in the subsequence of the "old" step is deactivated and the marker on the last step is moved to the empty container located on the same subworkspace as the subsequence.

When a marker is moved, new rules will be activated. These rules perform, in principal, the following actions:

1. give the attribute time-of-enter of the step which recently has received the marker the value of the current time,

2. deactivates the subworkspaces of all transitions connected directly after the step from where the marker recently came,

3. activate the subworkspaces of all transitions connected directly after the step to where the marker recently has been moved.

A rule that performs the actions mentioned above is shown first below. If the step marked by the marker is a structural step, the rules must take this into acount. The marker on the subworkspace containing the subsequence must be moved to the first step in the subsequence.

```
whenever any marker m is moved by G2
 and when the distance between m and the
    nearest container > the distance
    between m and the nearest step
 and the type of the step st nearest to m
    is terminal
 and there exists a transition connected at
    an output of st
 and not(there exists a parallell-bar
    connected at an input of st)
then conclude that the time-of-enter of the
    step nearest to m = the current time
 and deactivate the subworkspace of
    every transition connected at an
    output of the step connected at the
    previous-step of every transition
    connected at an input of the step
    nearest to m
 and activate the subworkspace of every
    transition connected at an output of st
```

(6)

### Restrictions

There are two restrictions concerning how steps may be interconnected. These restriction are to avoid additional and more complex rules. First, any step before or after a parallel transition must not be structured. Second, the final step in a subsequence must not be structured. The workaround for this is to include an empty dummy step where it is needed.

### Error handling

When an error occurs and the process is running, e.g. the temperature in the holding cell suddenly drops, certain actions must be carried out. There must be rules that make sure that the process is taken to a "safe" step and if possible make sure that certain sequences are performed before this "safe" step, e.g. emptying the production tank and cooling the plant. From this "safe" step, the process can be restarted after the error has been located and taken care of. The error can be located with the alarm analysis part of the control system (see Section 6.5). In most cases, these extra sequences are not included in the normal sequence net (because this would make the sequence net too muddled and the original structure of the net would be lost). To handle the sequence net when errors occur, the orginal generic rules are not sufficient. Therefore additionaly rules are needed.

We have chosen to implement the error handling in the case when the temperature in the

holding cell (under the production phase) drops below a given alarm limit, because this is the most illustrative case to show how the error handling works. When this alarm (TLS42) occurs, the following actions are taken:

1. Remaining product in the process is returned to the tank.

2. The plant is cooled.

3 The plant is stopped.

4 CIP (cleaning in place) must be performed before new production.

The actions 1 and 2 are a sequence that must be performed before the plant is stopped. This sequence is not included in the normal sequence net, but is found on a subworkspace of an object called production-error. This object in turn, is found on the subworkspace of the production step called p2-production (See Figure 6.9 and 6.10). In this subworkspace there are rules that take care of the sequence net in a proper way.

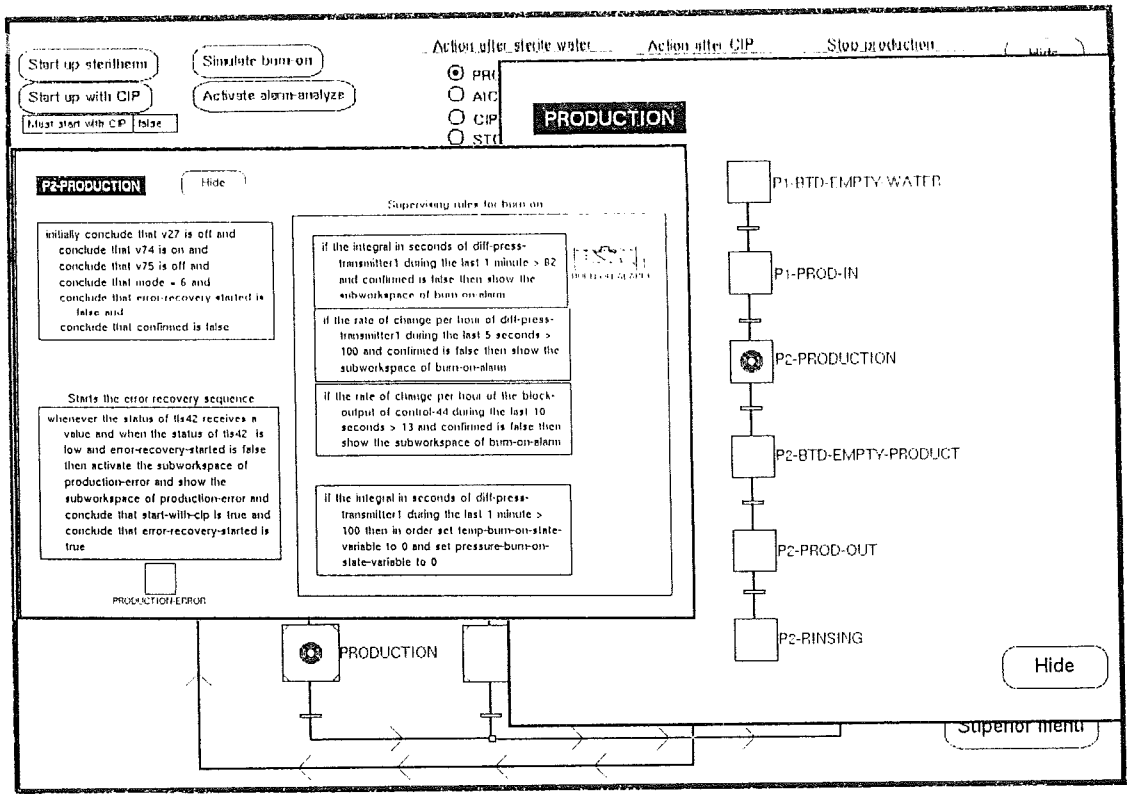



**Figure 6.9** The subsequence of the production step and the subworkspace of the p2-production step

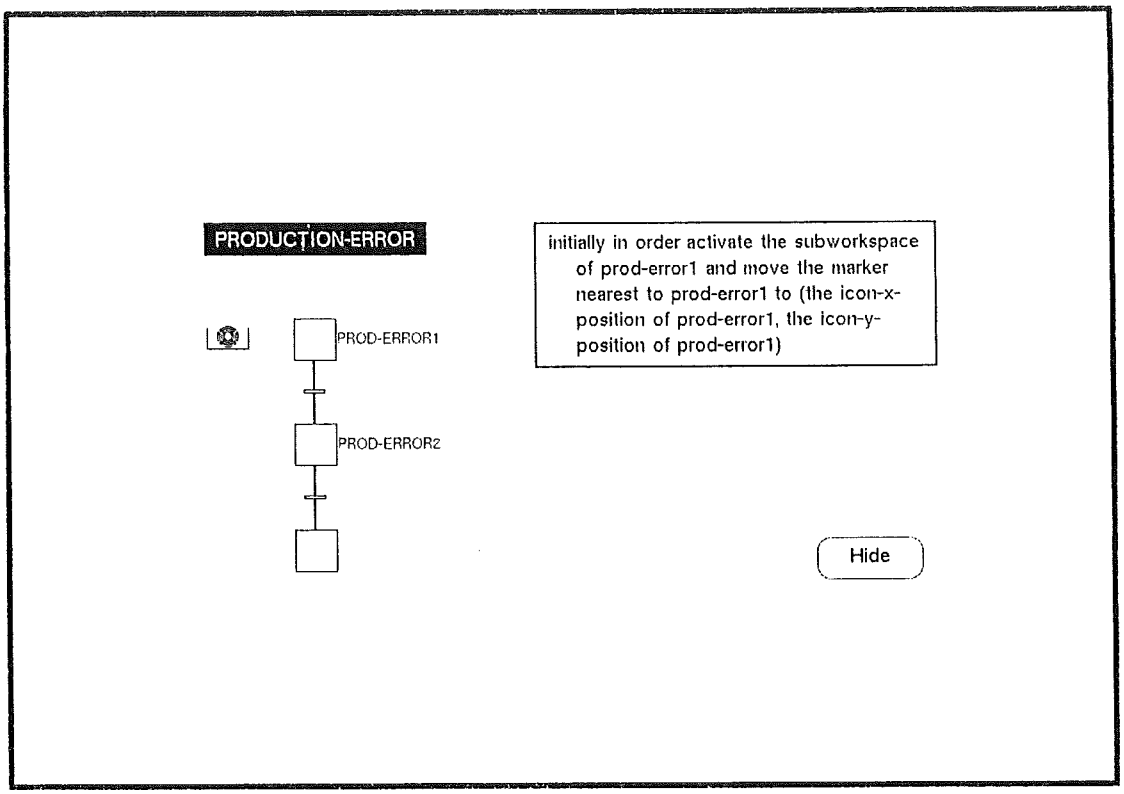



**Figure 6.10** This sequence is activated when an error occurs and the process runs in the production phase

In the subworkspace of the production step called p2-production (see Figure 6.9) there is a "whenever-rule" that fires when an alarm from TLS42 is given. This rule activates the subworkspace of the object called production-error where the error sequence is located (see Figure 6.10). There is an "initially-rule" on this subworkspace that activates the first step in the error sequence and moves the marker to the same step. When this rule has fired, the other rules that handle the sequence net can take over. The third and final step in the error sequence is not actually part of the sequence, but is needed to make the process stop and to move all markers in the sequence net to containers (except the marker in the main sequence, that is moved to the start step). In the third step there is an "initially-rule" that invokes three other rules. These three rules are found on the same workspace as the generic rules and perform the following actions

**Rule 1**: Moves all markers in the sequence net to containers.

**Rule 2**: Deactivates the subworkspaces of all steps and transitions in the sequence net.

**Rule 3**: Moves the marker of the main sequence to the start-up step.

When the process is started after an error, one must click on the button called "Start with CIP" on the operator's control. This will make the process start with a CIP (Cleaning in place) before it can

start normally.

## 6.5 The alarm analysis

This section describes the part of the control system that handles the alarm analysis. Alarms can arise when, e.g. the pressure in the holding cell is to low. The cause of this can be a broken pump and it is up to the alarm analysis part to guide the operator or service personnel in finding this error. The alarm analysis is found on the view called alarm-trees of the Steritherm object (see Figure 6.3 and Section 6.1 about the object-oriented model of Steritherm).

The alarm analysis starts when the button marked "Alarm analysis" on the operator's control is activated (see Section 6.4 about the operator's control). The alarm analysis works interactively with the user. Questions pop up when the system needs to know anything from the user and he can answer these questions by clicking on the appropriate buttons.

### Function

The alarm analysis is constructed as a fault tree with objects and a number of rules. The fault tree is the basis for fault localization. The origins of the fault tree implemented here is found in Appendix B. The fault tree is made up of a number of nodes, where the top node represents the alarm, e.g the temperature in the holding cell is too low, and where the leaf nodes represent either a physical fault, e.g. a broken pump rotator, or a functional fault, e.g. the steam boiler is not working. The rest of the intermediate nodes describe only functional faults. These functional faults differ from the functional faults in the leaves in the sense that it is possible to further encircle the fault from these nodes. Of the intermediate nodes there are two different kinds of functional faults.

In the first case, the system knows for sure that the fault must be found in the subtree of this functional node. A node like this can represent the fact that the pressure in the holding cell is to low. This in turn is a consequence of a question to the user, e.g. "is the pressure in the holding cell low ?". By knowing this, it is possible to prune all other parts of the tree and only concentrate on this subtree. If it turns out that no fault could be found, this does not mean that the fault may be found in another part of the tree. Instead this subtree has to be reconstructed to be able to locate more faults.

The second type of functional faults in the intermediate nodes are only used to divide the analysis into smaller and smaller parts. In these nodes, the system can not know if the fault is in the subtree of a node like this. The system tries with backward chaining, to verify the fault hypothes of the node. If it fails, the system will try to locate the fault in other nodes in the rest of the fault tree.

To verify if a specific leaf node represents the fault or not, a question is asked to the user. In the question, the possible fault is described. If the answer is positive, the fault analysis is over

otherwise the analysis will continue. The order in which the nodes are examined is determined by depth-first traversal. The nodes are arranged so that the nodes representing the most probable fault are examined first.

The questions in the leaf nodes differ from the questions in the intermediate nodes. The questions in the intermediate nodes are used to lead the system in the right direction and with that exclude the other parts of the tree. The questions in the leaf nodes are used to verify if the node represents the actual fault or not.

## Implementation

The alarm analysis is implemented with fault objects and rules. The objects represent different nodes in the fault tree, i.e. alarm, functional fault, and physical fault. The objects are connected to form a tree. The rules handle the search of a fault, ask questions to the user, and take the appropriate action after an answer has been given.

### Fault objects

There are three different kinds of objects concerning the alarm analysis. The classes defining these objects and the variables that belongs to the alarm analysis are found on the top level workspace called alarm-tree-classes. The objects are used to represent alarm nodes, functional fault nodes, and physical fault nodes. These objects only differ in the icons (icons with an A represent Alarm nodes, icons with a F represent Functional fault nodes, and icons with a P represent Physical fault nodes). Figure 6.11 shows a part of the fault tree implemented in the system. The objects in the fault tree contain different kinds of information regarding the fault they represent. The objects contain the following attributes: status, ask-question, check, and explanation-workspace.

* **Status**: Is of the type depth-first-symbolic-variable. This attribute is used to indicate if the object with this attribute represents the actual fault. The attribute accepts the following three different values: "fault", "no-fault", and "cant-find-fault". "Fault" indicates that the object represents the actual fault. "No-fault" indicates the opposite. "Cant-find-fault" indicates that the fault could not be found in one of the leaves in this objects subtree. The rules can either forward chain or backward chain to other rules with help of this attribute.

* **Ask-question**: Is of the type depth-first-logical-variable. This attribute is used by the rules when a question is asked to the user.

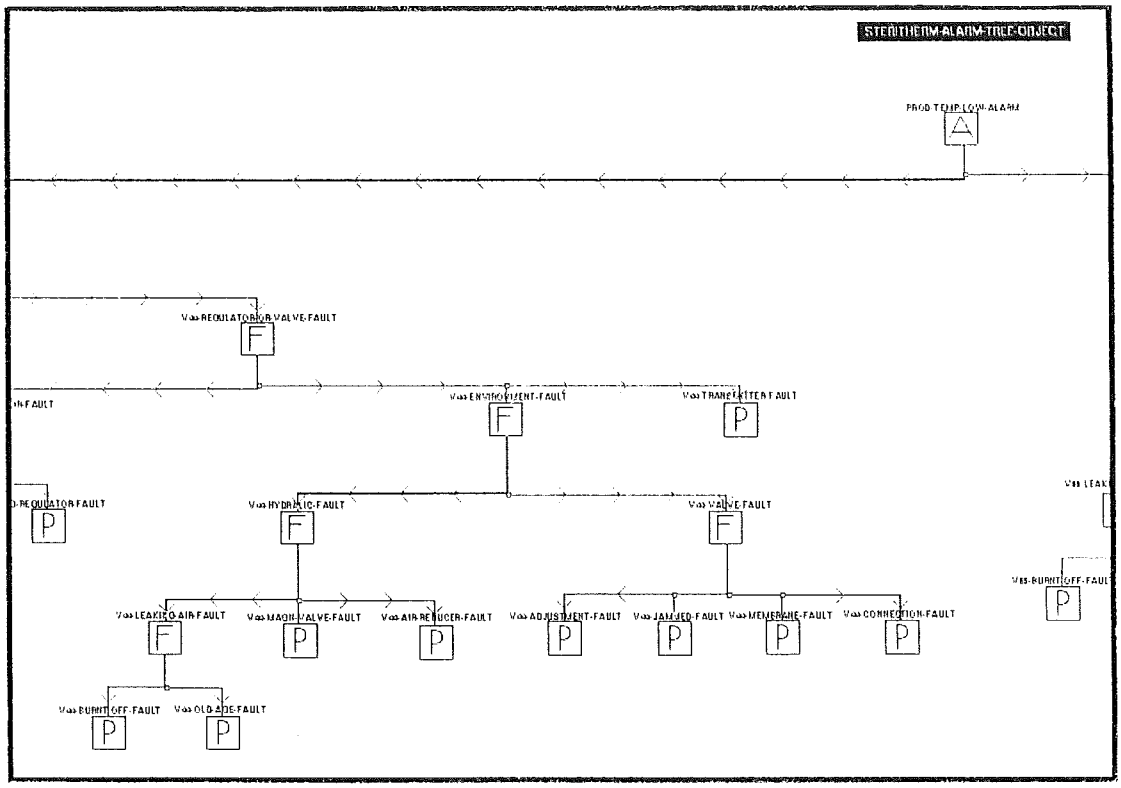



**Figure 6.11** A small part of the fault tree. The nodes are objects that represent various faults

* **Check**: Is of the same type as ask-question. This attribute is used by the rules when the system waits for an answer from the user.

* **Explanation-workspace**: This attribute is a pointer to a workspace that contains a textual explanation of the fault.

The subworkspace of the objects contain the question that is asked to the user. In addition to the questions there are also buttons to make it possible for the user to answer the questions. These buttons set the attribute check to "true" or "false" depending on if the answer is yes or no.

The variable type depth-first-symbolic-variable can only take symbolical values. It allows both forward chaining and backward chaining. This variable type has a validity interval of 1 hour. Depth-first-logical-variable can take logical values. It only allows backward chaining. This type also has a validity interval of 1 hour. The reason why the validity interval is 1 hour instead of indefinite is that after a time (hopefully greater than 1 hour) the control system may have to make a new alarm analysis and the old values of the attributes must not be left.

### Rules

The rules are used to control the whole fault localization procedure. Both backward chaining and forward chaining are used. The rules are all of the type

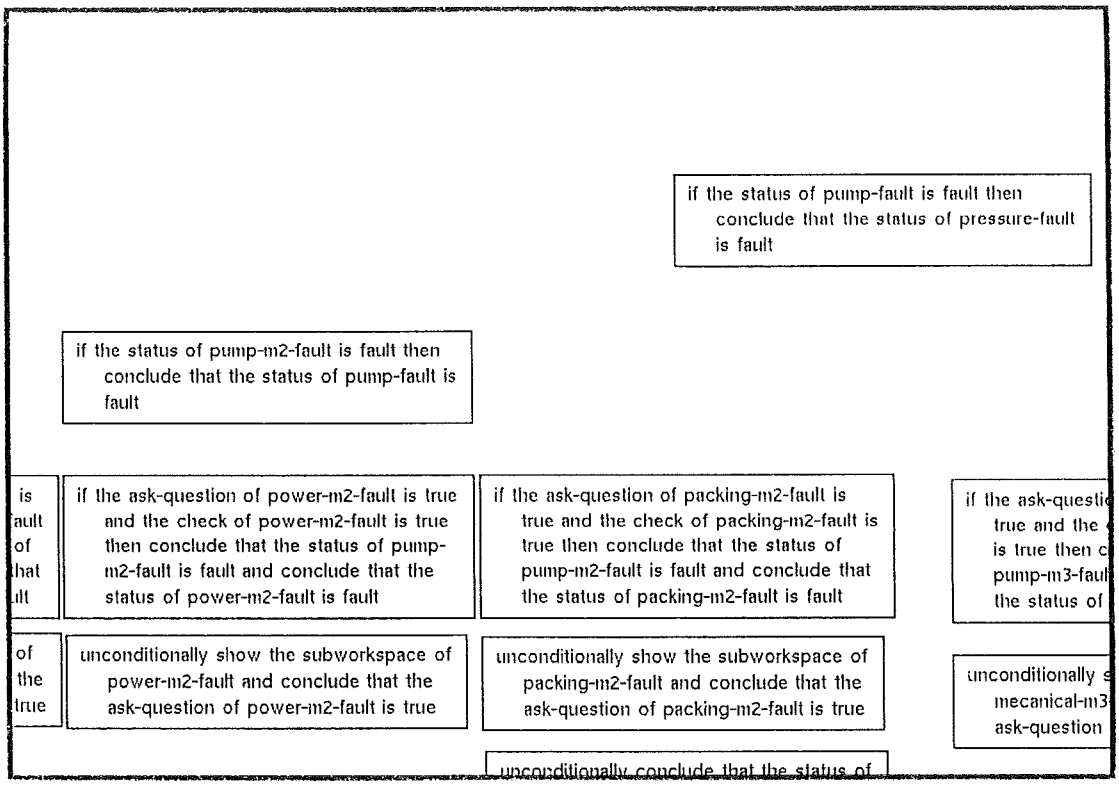



**Figure 6.12** A small part of the rules that handle the alarm analysis

**if** <conditions> **then** <conclusion>

or

**unconditionally** <action>

The rules are found on the subworkspace of the object called prod-temp-low- alarm-rules and this object is found on the same workspace as the fault-tree is located. The rules are geographically positioned similar to the fault objects. Figure 6.12 shows a small part of the rules.

The alarm analysis starts when the user clicks on the button marked "activate alarm analysis" on the operator's control. This button activates the top rule by using the focus action. The top rule look like this.

```
if the status of prod-temp-low-alarm has a
    value then conclude that the dummy is dum
```

(7)

This rule backward chains on the status attribute of prod-temp-low-alarm to the rules that have status of prod-temp-low-alarm in their conclusion parts. To make sure that the rules are tried in a certain order it is necessary to enter an integer value in the attribute depth-first-backward-chaining-precedence in the table of the rules. The rule with the lowest value will be visited first and the rest will be visited in increasing order. In the table of the rule, the option attribute must have the value "backward-chaining" to be able to backward chain at all. It must be noted that none of these

rules used in the alarm analysis have a scanning interval, because all rules are invoked either via forward or backward chaining. In rule 7 above, the conclusion part may be a bit confusing. The reason for this is that G2 does not allow an empty conclusion part of a rule. Some rules, like the rule above, are only used to backward chain to other rules and this means there is nothing to state in the conclusion parts.

Rule 7 backward chains to three different types of rules. The first type is a rule that in turn backward chains to other rules. Rules of this type divide the problem into smaller and smaller parts until a leaf rule is reached. This type of rule is shown below.

```
if the status of pump-fault is fault then
    conclude that the status of pressure-fault
    is fault
```
(8)

The second type of rule that rule 7 can backward chain to, may look like the following rule.

```
if the ask-question of pressure-fault is true
    and the check of pressure-fault is true
    then conclude that the status of prod-
    temp-low-alarm is pressure-fault
```
(9)

This type of rule is used when a question to the user must be asked. The question pops up when this rule in turn backward chains on the attribute ask-question of pressure-fault to a rule that looks like as follows.

```
unconditionally show the subworkspace of
    pressure-fault and conclude that the ask-
    question of pressure-fault is true
```
(10)

When the question has appeared, rule 9 will wait until check receives a value. When the user clicks on one of the buttons on the subworkspace with the question, check will receive either "true" or "false" depending on the answer. It is very important that the attribute time-out-for-rule-completion in the table of rule 9 has the value "none". This means that the inference engine in G2 never gives up trying to complete the rule. If the attribute for example had a value of "1 minute", the inference engine could have completed the rule (9) after 1 minute even if check had not received a value. When check has reveived the value "true" the rule completes and the conclusion part is executed. The inference engine now forward chains on the attribute status of prod-temp-low-alarm to the following rule.

```
if the status of prod-temp-low-alarm is
    pressure-fault then focus on pressure-
    fault and inform the operator that "There
    is a pressure fault"
```
(11)

This rule corresponds to a node in the fault tree where it is already established that the fault must be in one of the nodes in the subtree of this node. Even if the fault can not be found in the subtree it is not possible to find the fault in the rest of the fault tree. When it is established that the fault can be found in the subtree of this rule, the backward chaining is stopped at this point and rule 11 may forward chain (by using focus) to another rule that represents the top rule of the subtree. This rule is stated below.

```
if the status of pressure-fault is fault then
    conclude that dummy is dum
```
(12)

This rule backward chains on the attribute status of pressure-fault to the rules that have status of pressure-fault in their conclusion parts. From here the alarm analysis continues in the same way as from rule 7.

The third type of rules that rule 7 or rule 8 can backward chain to corresponds to a leaf in the fault tree. A rule of this type is shown below.

```
if the ask-question of mechanical-m2-fault is
    true and the check of mechanical-m2-fault
    is true then conclude that the status of
    pump-m2-fault is fault and conclude that
    the status of mechanical-m2-fault is true
```
(13)

This rule acts like rule 9, i.e. it backward chains to a rule like rule 10 that displays a question and then it waits for an answer. Unlike rule 9, this rule does not forward chain to another rule because this rule corresponds to a leaf in the fault tree. When the condition part is fulfilled, the fault has been localized and the alarm analysis is finished. The status attribute of the object (mechanical-m2-fault in this case) that this rule represents will get the value "fault" when the conclusion part executes.

In addition to those rules that have been described above, there is also one type of rule that can not be associated with a node in a fault tree. This rule must be used to make the alarm analysis work properly. A problem arises when rules of type 8 or 9, backward chain to a number of other rules. If all these rules fail, i.e the condition part of these rules is false, then the condition part of the rule that started the backward chaining (rule 8 or 9) will not receive any value at all (because the fired rules all failed). Since this rule has the value "none" in the attribute time-out-for-rule-completion, it will keep on trying to get a value for the condition part of the rule. This means that

the rule will backward chain again to the rules that recently were examined. It does not help to change the value of the attribute time-out-for-rule-completion. The solution to this problem is to add an additional rule that looks like the following.

```
unconditionally conclude that the status
    of pump-m2-fault is no-fault
```
(14)

This type of rule is positioned under all rules that perform backward chaining. It is necessary that this rule is visited last. This is accomplished by giving the attribute depth-first-backward-chaining-precedence the highest value. This means that the condition part of the rule one level up, that performed the backward chaining, now receives a value and that the inference engine will go on to the next rule.

## 6.6 Burn-on supervision

To get a high utilization of the process, it is important to supervise the burn-on effectivly. Earlier, the operator had an important role in the supervision of the burn-on. The operator estimates, depending on the product, when it is time for cleaning. A better method is to continuously supervise the effect of the burn-on and with this information, decide when it is time for cleaning.

The supervision of the burn-on is done in two different ways. First, the pressure difference on the product side of the heat exchanger I is measured. Figure 6.13 shows a diagram of the differential pressure due to the burn-on in the heat exchanger I. Second, the temperature increase of the heating water of the heat exchanger I, is measured. The pressure difference is measured by a differential pressure transmitter connected at the product side. The temperature of the heating water can not be measured directly because there is no temperature transmitter available. Instead, the output signal from the regulator PI-44 is used. This is possible because the increase in this signal is proportional to the increase in the temperature of the heating water in the heat exchanger I.

The supervision is performed by three rules that continuously check the signals from the transmitter DP and the output signal from PI-44. These rules are found in the production step (p2-production in Figure 6.9). The reason to put the rules here is that the rules shall only be active in the production phase. The rules are shown below.

```
if the integral in seconds of diff-press-
    transmitter1 during the last 1 minute > 82
    and confirmed is false then show the
    subworkspace of burn-on-alarm
```
(15)

if the rate of change per hour of diff-press-transmitter1 during the last 5 seconds > 100 and confirmed is false then show the subworkspace of burn-on-alarm (16)

if the rate of change per hour of the block-output of control-44 during the last 10 seconds > 13 and confirmed is false then show the subworkspace of burn-on-alarm (17)

The first two rules supervise the differential pressure. These rules handle the primary supervision. Rule 1 supervises the mean value of the differential pressure under a specified amount of time. When this mean value raises above a given limit, an alarm is given. Rule 2 supervises the rate of change of the differential pressure. This rule tries to detect fast changes in the differential pressure. If the rate of change is too big, an alarm is given. The last rule supervises the output signal from the regulator PI-44. This rule also uses the rate of change to detect an alarm.

When an alarm from the rules is given, a warning message appears on the screen. The user must click on a button to confirm the message. When this is done the the operator's control appears and the user can now stop the production and make a AIC or CIP.

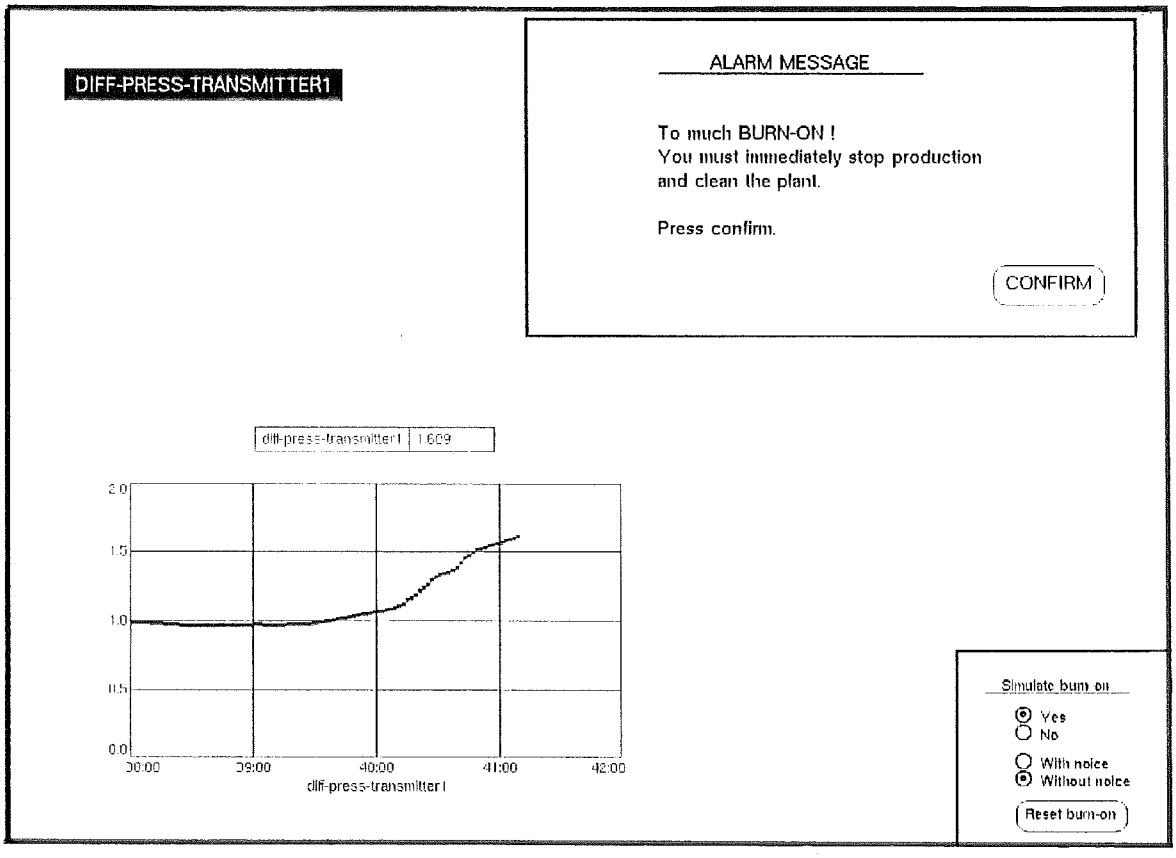



**Figure 6.13** Diagram showing how the differential pressure in a heat exchanger increases due to the burn-on

# 7 Experiences of G2

In general, G2 has been very nice to work with, when constructing the system described in this report. It would hardly be possible to construct this system with some other real-time expert system tools on the market today. G2 is probably the best tool for this kind of applications. The knowledge-base in G2 containing the implemented system is very large. It makes use of as much as 0.5 Megabyte on the hard disc. The knowledge-base consists of a total of nearly 4000 items (including objects, rules, and simulation formulas). It is probably one of the largest G2 applications constructed. The simulation part of the system is very time consuming, mainly because of the temperature equations in the heat exchangers. Due to the time consuming simulation, the system runs about three time slower than it is supposed to run. This means that, when the system "thinks" that one minute has past, the actual real time that has past is 3 minutes.

Inspite of the large knowledge-base and the time consuming simulaton, the system has run rather nicely and smoothly. However under the development of the system, some deficiencies of the G2 system have been noticed. The advantages have of course also been noticed. In the rest of this chapter we will discuss some of G2:s advantages and disadvantages that have been identified under the development of the project.

## Advantages

* **Generic simulation formulas and generic rules**: Makes it possible to easily change (e.g. add a new pump in the process schematic) the knowledge-base whithout the need for changing or adding more formulas or rules.

* **Referring to objects through connections**: The possibility to refer to objects through connections gives a very flexible system.

* **Very easy to work with**: G2 has a very nice graphics based development environment with windows, pop up menus, and mouse interaction. It is very easy to move, copy, and create parts of the knowledge base.

* **Editing**: The editor used to enter new rules and formulas or to edit old rules and formulas is very good.

* **Flexible rules and formulas**: When constructing rules and formulas you have many possibilities to construct what you want.

* **Hypermedia style interface**: The use of buttons, and windows, gives a system well

suited for rapid prototyping of e.g. user interfaces.

## Disadvantages

* **The manual**: It is very easy to learn the basic parts of G2, but when you need to find out how things really works, the manual tells almost nothing. Some kind of reference manual would be preferable.

* **Object names**: It is a great drawback in large systems with many objects of the same kind, not to be able to have the same name on two or more objects when they are positioned on different workspaces. It would then be possible to refer to them by using the name of the workspace where the object is positioned.

* **Difficult to control the inference engine**: Because G2 is a real-time expert system, it is not possible, in a controlled way, to stop at a rule and wait there until the user has typed in a value for a variable. The inference engine will forward or backward chain on other variables in the rule if it is possible. There should be some way of controlling this problem, maybe by some attribute in the rules table which states that the inference engine shall not go on until a specified variable has received a value.

* **Simulator**: Sometimes the simulator seems to be a bit unstable, i.e. it behaves very funny on some occasions. During the development of the simulation model many unexplicable internal errors and other unexplicable situations occurred. There are also some obvious bugs that occur e.g. when the simulator takes the square root of a variable that happens to be negative.

* **Limited number of datatypes**: There are only ordinary variables and constants. It would have been nice to be able to create our own datatypes, e.g. lists.

* **Small possibilities to show data to the user**: Particularly the graphs should be much more flexible.

* **Procedures**: Possibilities to write sequential statements in a procedure or function would be useful.

* **Icons**: An icon editor would have saved a lot of time.

* **Stubs**: It should be possible to declare some kind of general stubs of an object so one

could connect any kind of connection to this stub.

* **Printing of the knowledge base**: There should be a way to format the printing of the knowledge base. The way it works now, everything appears on the printing in a very muddled way.

* **Limited support for hierarchical objects**: Hierarchical objects, i.e. objects that are composed out of other objects are not completely supported by G2. It is possible to have objects as the values of object attributes, It is, however, not possible to simultaneously have graphical icon representations for these objects. Furthermore, the connection relation is not defined between objects at different workspaces and it is not possible to make explicit references between workspaces. An example of the latter is that it is not possible for an equation or rule on the subworkspace of an object to refer to that object.

* **Unique objects**: An object in G2 is unique. It is not possible to have two representation, e.g. icons for a single object. Even if the two objects are given the same name they are treated by G2 as two individual objects. In the common knowledge is it necessary that an object can occur at several places possibly with different graphical representations depending on the context or the representation that the object occurs in.

In spite of the disadvantages that has exist, it is probably safe to say, that it would have been totally impossible to implement a comparable system in any other programming environment, within the scoop of a master thesis project.

# 8 Conclusions

In this project, a knowledge-based real-time system for controlling the Steritherm process has been developed. To test the control system, a simulation model of the Steritherm process has also been implemented in the system. The system contains in principal all parts that were specified in the project specification (Årzen 1989).

The control system can run against the simulation model and it works as expected. With a few modifications the simulation model can be exchanged for the real Steritherm process when it becomes available.

The simulation model simulates pressures, flows, temperatures and levels in tanks. The temperature equations in the heat exchangers have been implemented with dynamic state equations. A dynamic model by Åström, 1989 has been used for these equations. The burn-on in the heat exchangers is simulated in two different ways. Both the temperature and the pressure in the heat exchangers are affected. It is easy to extend the simulator to also simulate different fault conditions.

The control system is the most important and interesting part of the project. This part consists of the following parts: hierarchical representation and views of the process, sequences, alarm analysis, regulators, and burn-on supervision.

The hierarchical representations represent the process at different degrees of resolution. Views are used to represent different kinds of knowledge at the different levels.

The sequence net handles the many different modes of the Steritherm process. The sequence net is based on the Grafcet formalism. In addition to the normal phases, the system also handles situations when a fault in the process appears. There are two PID regulators used to control the temperature in the holding cell of the process and the temperature of the water in the water tank. The alarm analysis is used to localize the cause of an alarm. It is implemented with both objects that represent faults and rules that control the analysis. The supervision of the burn-on is implemented with rules that supervise both the differential pressure and the water temperature in the heat exchanger.

The system is constructed in G2, a real-time expert system for process control. Even if it is possible to run the control system it is by far not complete, but with further development of the control system it may be something useful in the future.

# References

Årzen, K-E. (1989): "Knowledge-base control systems: A G2 prototype, Examensarbets-beskrivning", unpublished paper.

Årzen, K-E. (1989): "Steritherm - views and hierarchical levels", unpublished paper.

Åström, K.J. (1974): "Ventiler och Pumpar", Technical report TFRT-3110, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Åström, K.J. (1987): "Implementation of PID regulators", Technical report TFRT-7344, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Åström, K.J. (1989): "A Dynamic Model for the Heat Exchangers in STERITHERM", unpublished paper.

IT4 report (1988): "Knowledge-Based Real-Time Control Systems, IT4 feasability study".

Lannefors, L. (1987):"Funktionsdiagrammet Grafcet", Liber läromedel, Stockholm.

Activation-chart STERITHERM P-SATT 4 EKSTROEM , SVERIGE U074140 FLOWCHART:30245-0431

PAGE 1 (2)

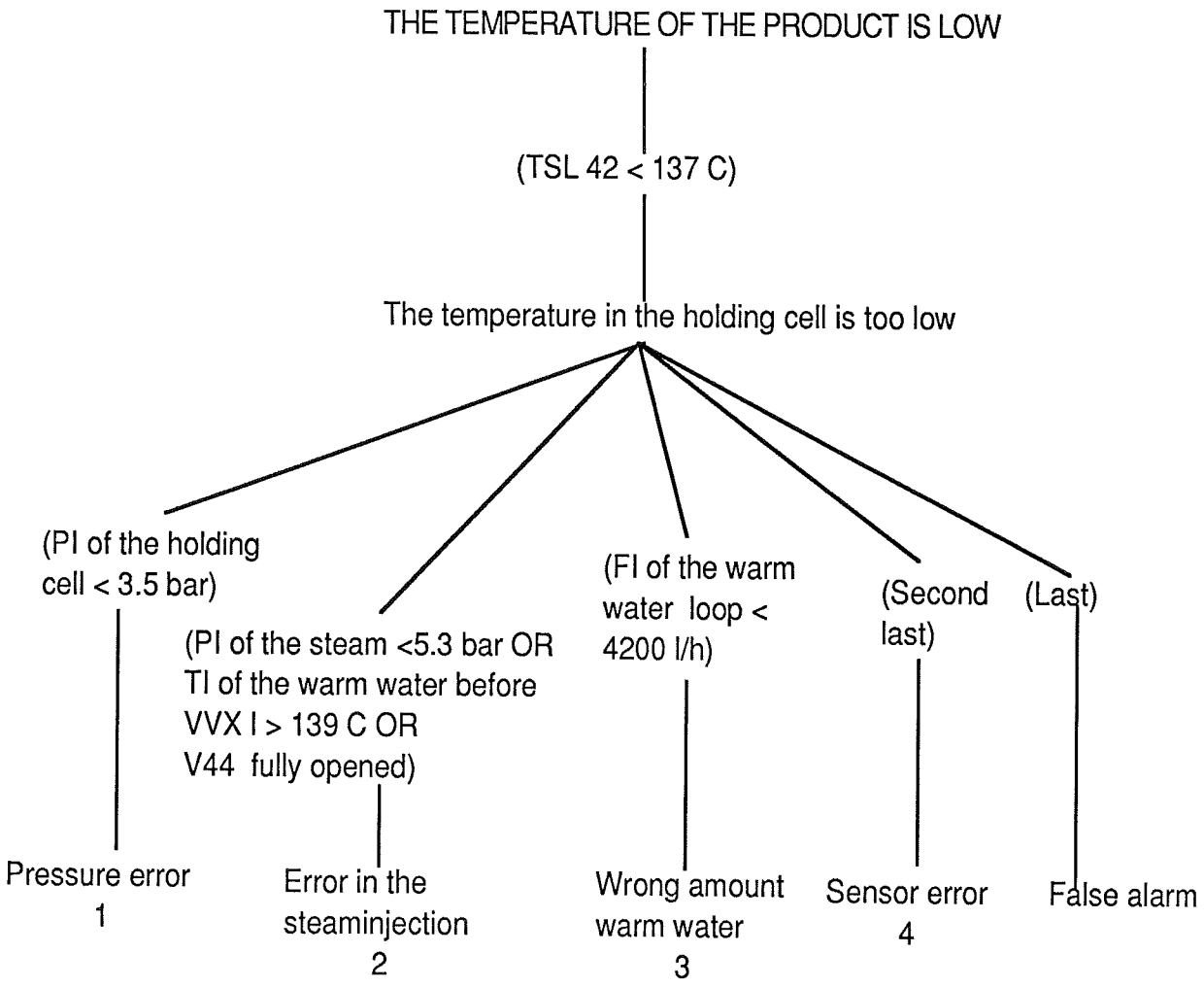
| STEPNAME | STEP | COND | M2 | M3 | M9 | M11 | M12 | M13 | V11 | V13 | V14 | V20 | V27 | V44 | V61 | V64 | V66 | V71 | V73 | V74 | V75 | V78 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STERILE WATER | | | | | | | | | | | | | | | | | | | | | |
| HEATING | 112 | TS71 | X | X | X | | | | | | X | A | | X | A | X | ‡ | X | X | | X | |
| STERILIZ | 112 | T 1 | X | X | X | | | | | | X | A | | X | A | X | ‡ | X | X | | X | |
| COOLING 1 | 113 | T 2 | X | X | X | | | | | | X | ‡B | | X | ‡B | X | ‡ | X | X | | | |
| COOLING 2 | 114 | T 3 | X | X | X | | | | | | X | | | X | ‡B | X | ‡ | X | X | | | |
| STERILE W | 115 | | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | X | X | X |
| | PRODUCTION | | | | | | | | | | | | | | | | | | | | | |
| BTD EMPT | 117 | LS65 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | X | | X |
| PROD IN | 118 | T 4/5 | X | X | X | X | | | | X | | | | X | | X | ‡ | X | | X | | X |
| PRODUCTION | 119 | | X | X | X | X | | | | X | | | | X | | X | ‡ | X | | X | X | X |
| BTD EMPT | 120 | LS65 | X | X | X | X | | | | | | | | X | | X | ‡ | ‡ | | X | | X |
| PROD OUT | 121 | T 6/7 | X | X | X | X | | | | | X | | | X | | X | ‡ | ‡ | | X | | X |
| RINSING | 122 | T 8 | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | X | | X |
| STERILE W | 115 | | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | X | X | X |
| | ASEPTIC INTERMEDIATE CLEANING | | | | | | | | | | | | | | | | | | | | | |
| RINSING | 132 | | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | | | X |
| CAUSTIC DOS | 133 | T11/12 | X | X | X | X | X | | | | X | | | X | | X | ‡ | X | | | | X |
| CAUSTIC CIRC | 134 | T 13 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | | ‡ | X |
| BTD EMPT | 135 | LS65 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | | X | X |
| RINSING | 136 | T 14 | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | | | X |
| | STOP | | | | | | | | | | | | | | | | | | | | | |
| BTD EMPT | 142 | LS65 | X | X | X | | | | | | | | | | | X | ‡ | X | | | | X |
| RINSING | 143 | T 21 | X | X | X | | | | | | X | | | | | X | ‡ | X | | | | |

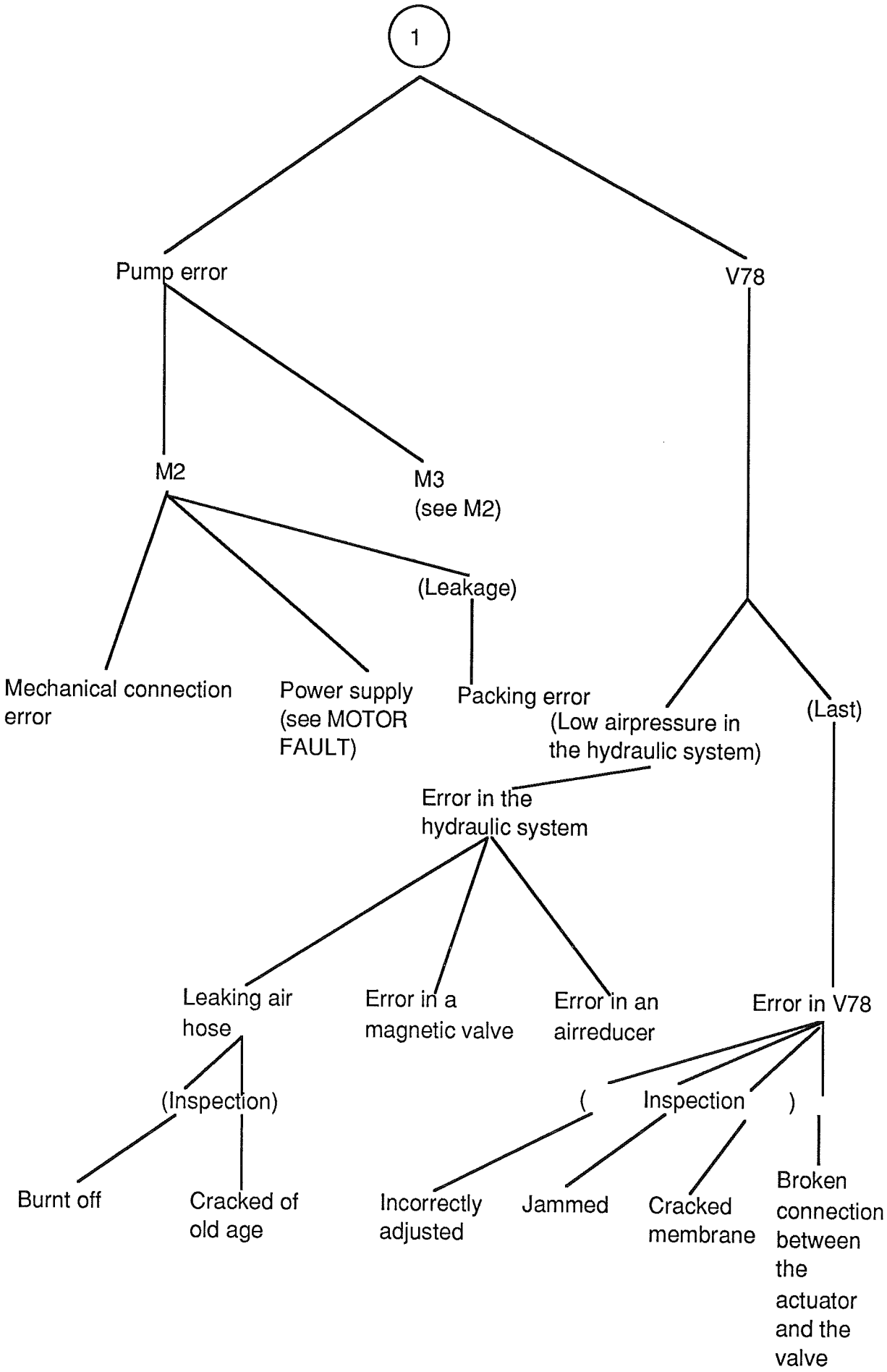
| STEPNAME | STEP | COND | M2 | M3 | M9 | M11 | M12 | M13 | V11 | V13 | V14 | V20 | V27 | V44 | V61 | V64 | V66 | V71 | V73 | V74 | V75 | V78 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

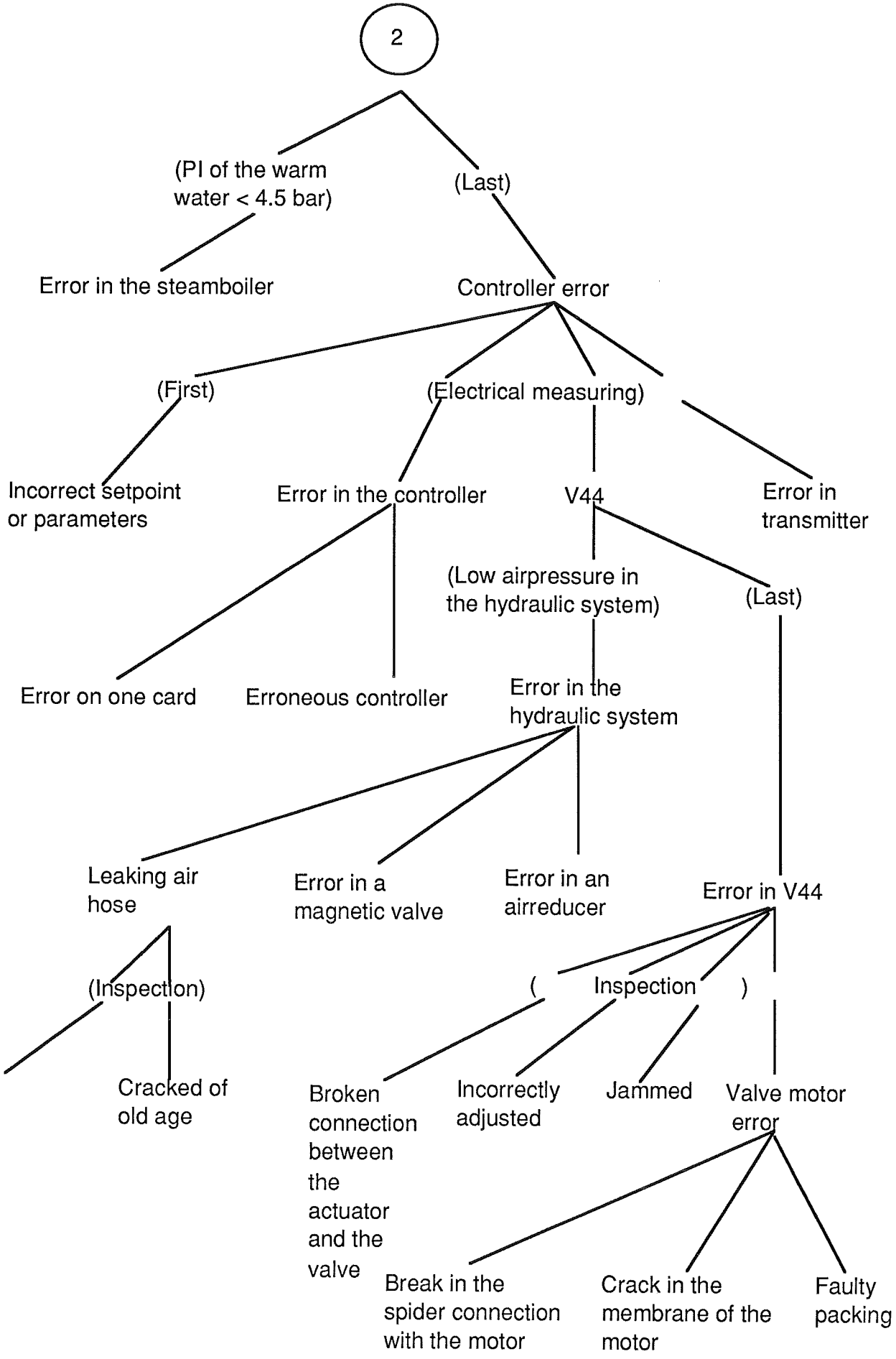
| STEPNAME | STEP | COND | M2 | M3 | M9 | M11 | M12 | M13 | V11 | V13 | V14 | V20 | V27 | V44 | V61 | V64 | V66 | V71 | V73 | V74 | V75 | V78 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLEANING | | | | | | | | | | | | | | | | | | | | | |
| RINSING | 152 | | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | | | X |
| CAUSTIC DOS | 153 | T31/50 | X | X | X | X | X | | | | X | | | X | | X | ‡ | X | | | | X |
| CAUSTIC CIRC | 154 | T 33 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | | X | X |
| BTD EMPT | 155 | LS65 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | | | X |
| LEVEL ADJ | 156 | T 34 | X | X | X | X | | | | | X | | | X | | X | ‡ | X | | | | X |
| CIRC V13 | 157 | T 35 | X | X | X | X | | | X | | | B | | X | | X | ‡ | X | | | X | X |
| BTD EMPT | 181 | LS65 | X | X | X | X | | | | | | | | X | | X | ‡ | X | | | | X |
| RINSING | 158 | T 36 | X | X | X | X | | | | | X | | | X | B | | ‡ | X | | | | |
| ACID DOS | 169 | T45/51 | X | X | X | | | X | | | X | | | X | B | | ‡ | X | | | | |
| ACID CIRC. | 170 | T 46 | X | X | X | | | | | | X | | | X | B | | ‡ | X | | | X | |
| BTD EMPT | 171 | LS65 | X | X | X | | | | | | | | | X | B | | ‡ | X | | | | |
| RINSING | 168 | T 44 | X | X | X | | | | | | X | | | X | B | | ‡ | X | | | | |
| STEPNAME | STEP | COND | M2 | M3 | M9 | M11 | M12 | M13 | V11 | V13 | V14 | V20 | V27 | V44 | V61 | V64 | V66 | V71 | V73 | V74 | V75 | V78 |

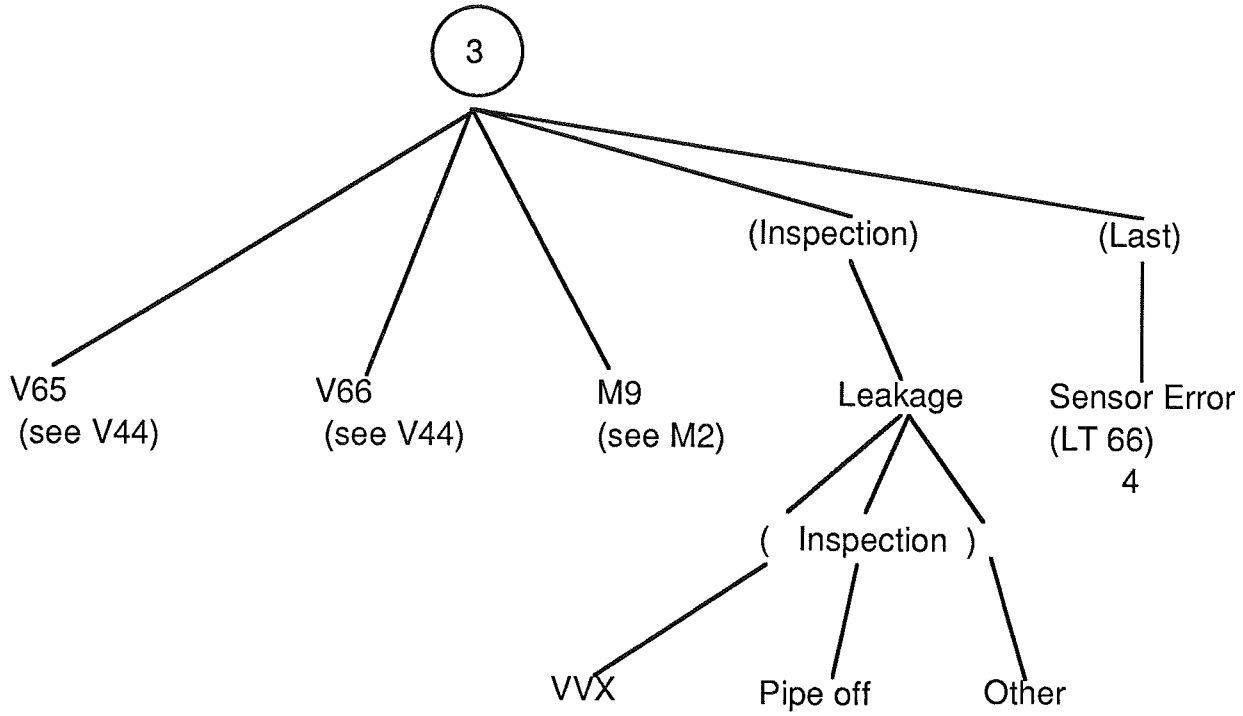
‡ = ACTIVATED ACCORDING TO SUBPROGRAM OR LEVELSENSOR

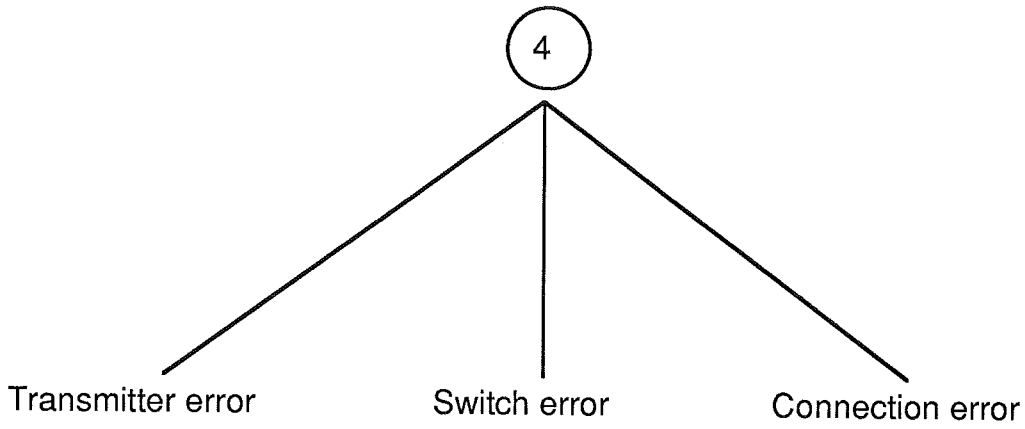
# B. Search tree for fault diagnosis




1
Pump error
V78
M2
M3
(see M2)
(Leakage)
Mechanical connection error
Power supply (see MOTOR FAULT)
Packing error
(Low airpressure in the hydraulic system)
(Last)
Error in the hydraulic system
Leaking air hose
Error in a magnetic valve
Error in an airreducer
Error in V78
(Inspection)
(
Inspection
)
Burnt off
Cracked of old age
Incorrectly adjusted
Jammed
Cracked membrane
Broken connection between the actuator and the valve


2
(PI of the warm water < 4.5 bar)
(Last)
Error in the steamboiler
Controller error
(First)
(Electrical measuring)
Incorrect setpoint or parameters
Error in the controller
V44
Error in transmitter
(Low airpressure in the hydraulic system)
(Last)
Error on one card
Erroneous controller
Error in the hydraulic system
Leaking air hose
Error in a magnetic valve
Error in an airreducer
Error in V44
(Inspection)
(
Inspection
)
Cracked of old age
Broken connection between the actuator and the valve
Incorrectly adjusted
Jammed
Valve motor error
Break in the spider connection with the motor
Crack in the membrane of the motor
Faulty packing


3
V65
(see V44)
V66
(see V44)
M9
(see M2)
(Inspection)
Leakage
( Inspection )
VVX
Pipe off
Other
(Last)
Sensor Error
(LT 66)
4


4
Transmitter error
Switch error
Connection error