I-SS: Integrated Supervision Systems Approach based on Interactive Components

Orlando C. Contreras N.
Grup *eXiT* - IIiA
Universitat de Girona & LEA-SICA
Av. Lluís Santaló s/n, 17071, Girona
Spain
&
Laboratorio de Computo Especializado
Univeridad Autónoma de Bucaramanga
Calle 48 No. 39-234, Bucaramanga
Colombia

Josep Lluís De La Rosa
Grup *eXiT* - IIiA
Universitat de Girona & LEA-SICA
Av. Lluís Santaló s/n, 17071, Girona
Spain

Joaquim Melendez Frigola
Grup *eXiT* - IIiA
Universitat de Girona & LEA-SICA
Av. Lluís Santaló s/n, 17071, Girona
Spain

Abstract – **Process supervision is the activity focused on monitoring the process operation in order to deduce conditions to maintain the normality including when faults are present. Depending on the number/distribution/heterogeneity of variables, behaviour situations, sub-processes and so on from processes, human operators and engineers do not easily manipulate the information. This causes the necessity of automation of supervision activities. Nevertheless, the complexity to deal with the information difficult the design and development of software applications. We present the approach called "*Integrated Supervision Systems*". It proposes multiple supervisors, under co-ordination, should supervise multiple sub-processes whose interactions should permit to supervise the global process.**

I. INTRODUCTION

Process supervision is the branch of process control that takes into account the way in that a process operates, or changes, i.e., the way in that the process behaves. It implies to monitor the process operation by means of redundancy (additional sensors, knowledge and so on) in order to deduce conditions to maintain the normality including when faults (misbehaviours) are present [1] [2]. According to this goal, operators and engineers deal with the process information (variables, parameters and relations among them), reasoning on it in order to identify and diagnose faults and to propose correction actions. Depending on the number/distribution/heterogeneity of variables, behaviour situations, interacting sub-processes and so on (according to the process), that information is not easily to manipulate. This causes the necessity of automation of supervision activities. Nevertheless, those issues difficult also the construction of software applications.

Software applications specially constructed to automate supervision activities are so-called *supervision systems* (SSs). The goal is to reduce the dependency on human operators to assure the normal process operation. This kind of applications use process behaviour models (PBMs) that describe normal operation situations in order to detect deviations (faults), to deduce the origin (diagnose them) and to propose or to execute appropriated correction actions.

The construction of a SS involves an important analysis and treatment on information from a process in order to obtain a PBM that could be easily mapped into computational structures and easily developed as software. Information analysis and processing increases with the process complexity. Usually the process is so complex that is either difficult or inappropriate to describe all the situations into an unique and complete PBM. This is due mainly to the big volume of information (from a big number of variables and behaviour situations) and interactions of multiple distributed heterogeneous sub-processes (according to the process nature). As a consequence it is difficult to map a computational model that could be developed as software. We cope with that complexity with specialised components tuned to solve simple tasks that should operate under co-ordination, which are based on concepts of software agents [3] [4]. Software agents offer capabilities (encapsulation, problem-solving focused, autonomy, co-operation and so on) that allow to manage the complexity by dealing with the multiplicity/distribution/interaction/sharing of tasks and information. We propose that multiple PBMs from sub-processes[1] (sub-PBMs) might be obtained and integrated. Subsequently, multiple supervisors (and other components that assist the tasks) should be developed on those sub-models. Sub-PBMs should capture the sub-process behaviours whose interactions should capture the global behaviour; supervisors should supervise sub-processes whose interactions should supervise the global process. This proper approach is called "*Integrated Supervision Systems*".

In the following sections the proposed approach is briefly presented. General concept, components and models that define it are presented in the section II. Software platform developed to apply it is presented in the section III. Example to clarify it and conclusions are presented in the sections IV and V respectively.

II. INTEGRATED SUPERVISION SYSTEMS APPROACH

A. Introductory

The State of the Art about process supervision does not refer approaches to deal with the complexity of SSs and nor approaches to develop this kind of applications, particularly with interactive components. It has mainly treated the use of

[1] Sub-process refers part of process.

software techniques (of artificial intelligence and procedural programming) in the development of applications with the objective of automate supervision tasks and of provide support to human operators during the process operation [5] [6] [7] [8]. Investigations have also treated interactions and co-operations among applications motivated by the positive aspects of distributed processing performance, flexibility, modularity and resource sharing [9] [10] [11] [12] [13]. Nevertheless, greater part of results were closed solutions that did not take into account the heterogeneous and distributed nature of complex processes, process flexibility and multiplicity of tasks that must be tuned to achieve the supervision goals.

B. General Concept

An **integrated supervision system (I-SS)** is defined as a "*system with the ability to sense a process and act on it, composed of interactive components for reasoning about the process behaviour in order to propose (and to execute) appropriated actions to maintain the normal operating conditions in case of faults*". It is based on the procedure depicted in the Fig. 1. Process constitutes the environment with which components interact. It provides perceptions, and actions are exerted on it. Variable measurements and additional information from them constitute perceptions. Analysis functions applied on variable measurements obtain that additional information, which we name "*abstractions*". Set point changes, parameter re-tunes and so on constitute actions to execute on the process. Messages to humans, e.g., showing messages on screen, firing alarms and so on, constitute also actions.

In general, an I-SS is a set of different supervisors that should supervise multiple sub-processes of a process, whose interactions should permit to supervise the global process. Multiple sub-PBMs from the process behaviour might be obtained and integrated. Sub-PBMs should capture the sub-process behaviours whose interactions should capture the global behaviour. Subsequently, multiple components should be developed on those sub-models; components should accomplish activities to achieve the supervision goals. Fault detection, fault diagnosis and reconfiguration methods [1] [2] [14] should be used and combined to analyse situations that should compose sub-PBMs. Computational mappings from them constitute data and knowledge for components.

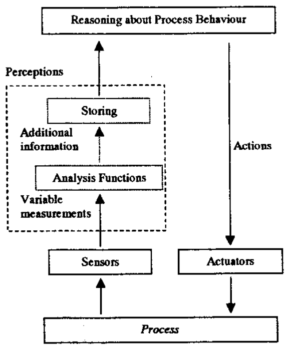


Fig. 1 Process information treatment

C. Components

1) Basic Components: They are focused on the treatment of the process information, i.e. on the acquisition, abstraction, storing, reasoning and execution of/on data and behaviour situations, according to the procedure depicted in the Fig. 1. Five kinds of basic components so-called *perceptors*, *actuators*, *abstractors*, *perception bases* and *supervisors* are defined. The behaviour (the way of acting) of a component is determined by services that it must support to an I-SS for dealing with the information and for interacting with process and humans. Interactions among those components and with process and humans are showed in the Fig. 2. Predetermined charges of those components are:

- **Perceptors** constitute mechanisms for inputting data from process. They perceive updated variable measurements and supply abstractors, perception bases and supervisors with that information. Devices (sensors) are linked to those components.
- **Actuators** constitute mechanisms for outputting data to process. Actuators execute actions (set point changes, re-tuning parameters and son on) on process. Devices (actuators) are linked to those components.
- **Abstractors** are in charge of abstracting information from acquired variable measurements. They elaborate, by means of analysis functions, significant information for interpreting updated variable facts, e.g. trends, deviations, mean values and so on.
- **Perception bases** are stores of (updated and historical) variable measurements and abstractions. That information indicates how process evolves through time.

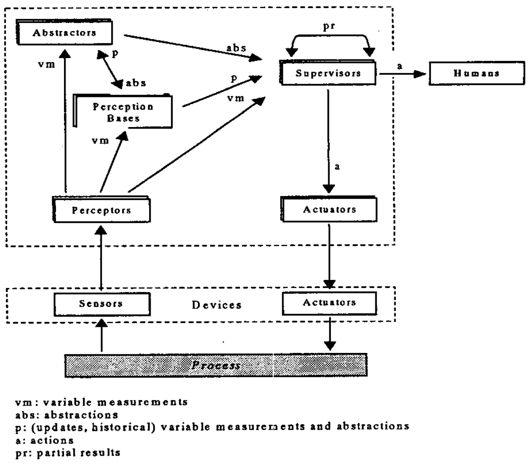


Fig. 2 Interactions among basic components, and with process and humans

- **Supervisors** are in charge of reason on the behaviour of process (or sub-processes). Fault detection, fault diagnosis and reconfiguration are tasks associated to those components. Action execution is achieved through actuators.

2) Support Components: They are focused on services to basic components. Two kinds of support components so-called *assistants* and *fact bases* are defined. The behaviour of a components is determined by services that it support to basic components for accomplishing their tasks. Interactions among basic components and support components are showed in the Fig. 3. Predetermined charges of those components are:

- **Assistants** are in charge of accomplishing support operations for abstractors and supervisors, e.g. mathematical operations.
- **Fact bases** are stores of happening facts, i.e., stores of partial results, actions and so on, which supervisors could need for accomplishing tasks.

3) Coordination Components: They are focused on the control/co-ordination of before mentioned components. Only a kind of those components so-called **co-ordinators** is defined. They are in charge of control basic and support components and of co-ordinate interactions among them. Interactions among co-ordinators and basic and support components are showed in the Fig. 4.

4) Operation Cycle among Components: When a process is operating, the I-SS perceives, by means of perceptors, updated variable measurements. Perceptors supply abstractors, perception bases and supervisors with that information. Abstractors access perception bases to get needed information to apply analysis functions on variable measurements. They supply perception bases and supervisors with the results. At that time, an abstractor could request support operations to one any assistant (if needed). Then, supervisors reason on the behaviour of the process (or sub-processes). They apply their knowledge on data (updated/historical variable measurements and abstractions). So, they detect and diagnose possible faults. They take decisions to cope with the detected faults (if exist). At that time, a supervisor could request support operations to one any assistant (if needed). It also could request partial results to one other supervisor (if needed). Then and depending on the decisions, supervisors send actions to actuators (or messages to humans). Moreover, they could supply facts into fact bases. Actuators execute the actions on the process. Also, at any time, any component could send the state of communication with some other component to the co-ordinator. Then, it reasons on that information and takes decisions on the interactions among the components. In that case and depending to the decisions, it sends communication actions to the involved components.

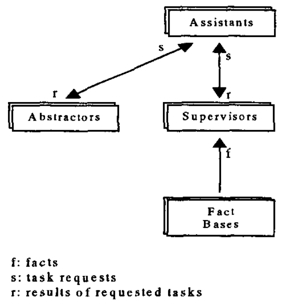


Fig. 3 Interactions among basic and support components

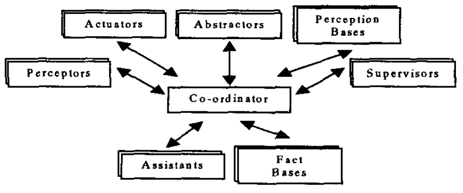


Fig. 4 Interactions between co-ordinator and other components

D. Models

1) Process Behaviour Models, PBMs: PBM captures situations of interest that determine the process behaviour through variables, parameters and relations (among them). If PBM refers a sub-process so, it is a sub-PBM. Real principles on process components, i.e. the general belief about the way that they would behave, and tasks to achieve the supervision goals must be taken into account to analyse situations.

Problem-solving techniques based on analytic/heuristic knowledge are used and combined for the interpretation of situations (more specifically of variables, parameters and relations that describe them). Analytic knowledge is used to produce quantifiable, analytical information from where process behaviour features could be extracted and then compared with normal operation features. Heuristic knowledge uses information in qualitative terms, which could be provided by process operators and engineers. In this way, variables, parameters, characteristic/generated values and so on can be represented as imprecise data, e.g. small, large, too hot and so on.

Three kinds of basic models are differentiated: *quantitative*, *qualitative* and *diagnostic*. Quantitative models describe the functional process behaviour with normally simple, analytic representations. Qualitative models describe the functional process behaviour with heuristic representations. Diagnostic models describe the functional process behaviour with pre-assigned links among symptoms, faults and actions. None kind of model is exclusive to describe determined process behaviours and a combination could be satisfactorily required according to problem-solving techniques used for the interpretation of situations.

The following example can clarify this: the sentence "*when the temperature in the reactor is too hot, open the input valve to 70-90%*" describes the situation "*the temperature experiments the behaviour of too hot*" that is happening in the "*reactor*", thereby necessitating an action "*open the input valve to 70-90%*" to cool it. Comparison of the measured temperature with a certain range of temperatures considered "*too hot*" should permit to determine if it is too hot. This situation is part of a sub-PBM that determine the behaviour of the "*reactor*", which is component of a plant.

2) Mapping PBMs: Techniques from artificial intelligence and procedural programming can be applied in different ways to map information from behaviour situations. They would be selected according to the techniques used for the analysis and interpretation of situations, e.g., sentences that describe situations in terms of qualities or if-then rules are normally mapped with artificial intelligence structures. None software technique is exclusive to problem-solving techniques and a combination could be satisfactorily required. In fact, the best strategy is using of all the available techniques (arrays, objects, heuristic rules, logic fuzzy and so on, or hybrid) to manipulate the information into computational structures. The use of one or other technique is only submitted to the treatment of data and situations, which would determine the I-SS complexity (imprecision, uncertainty, heterogeneity, distribution and volume of information).

For example, the sentence "*if TempReactor == TOO_HOT { action(OPEN_INPUT_VALVE, 80) }*" shows a mapping of the before mentioned situation. The symbol "*TempReactor*" represents the variable "*temperature in the reactor*"; the symbol "*TOO_HOT*" represents the range of temperatures considered too hot; the symbol "*OPEN_INPUT_VALVE*" represents the action "*open the input valve*"; the symbol "*80*" (which is in the range of 70-90) represents the measurement of opening the input valve that would be executed.

3) Interacting Components Models, ICMs: Interactions among components that compose an I-SS should be co-ordinated using an ICM, which captures situations of interest through "*communication acts*" (among them). Communication acts are special actions that a component accomplishes to interact with other ones, e.g. establishing communication, sending data, requesting a specified task and so on. A communication language, described in the section III, defines the communication acts.

Example to clarify this: the sentence "*when faulty communication with supervisor$_1$, stop it and establish communication with supervisor$_2$*" describes the situation "*the I-SS experiments a faulty communication*" that is happening with "*supervisor$_1$*", thereby necessitating the actions "*stop supervisor$_1$*" to that it stops activities and "*establish communication with supervisor$_2$*" to that it accomplishes activities that supervisor$_1$ was accomplishing.

4) Mapping ICMs: A communication protocol, described in the section III, specifies the communication acts how common communication patterns. These patterns are used to map interactions among components.

For example, the sentence "*if ComSupervisor$_1$ == TIME_OUT { Ungo(Supervisor$_1$); Go(Supervisor$_2$) }*" shows a mapping of the before mentioned situation. The symbol "*ComSupervisor$_1$*" represents the variable that registers the communication with supervisor$_1$; the symbol "*TIME_OUT*" is the response act that indicates communication faulty by excess time; the symbols "*Ungo*" and "*Go*" are execution acts that indicate stop and run a component respectively.

III. SOFTWARE PLATFORM

I-SS software platform provides the necessary frame to develop and to communicate the I-SS components. It is basically the software framework that provides the computational machinery both of the structures for components and of the infrastructure for communications. It is based and developed on the so-called "*Intelligent Control architecture*" (ICa)[2] [15], a framework to support the design and development of flexible and interoperable distributed software.

A. Component Structures

Component structures specify how components should be developed. Two basic structures were obtained[3], which are briefly described.

Components of perceptor, actuator, abstractor, supervisor, assistant and co-ordinator kinds share the agent structure showed in the Fig. 5. Execution mechanism permits to run/stop a component. Communication mechanism registers the component on the software platform[4] on which it runs, and permits the communication with other components through that platform (see the Fig. 7). Own data and operations constitute information and activities to support particular tasks that a component accomplishes. Aim is the core of the knowledge that a component has to accomplish tasks, to interact with other components and with its environment and humans in order to achieve own goals. It is inherent in component, and it is continuously executing when the component is running.

[2] The "Autonomous Systems Laboratory" (ASLab) developed ICa. It authorised to use ICa in researches and developments.
[3] The computational structures were developed to implementations in C++ language.
[4] The developed communication infrastructure permits to execute/interact components on Win32 platform.

Components of perception base and fact base kinds share the data-store structure showed in the Fig. 6. Execution mechanism permits to run/stop a data-store. Communication mechanism registers the data-store on the software platform on which it runs, and permits to put/get data into it.

B. Communication Infrastructure

1) Communication Language: I-SS communication language specifies the structure of messages that components use for communications. The fundamental view of messages is that they represent communication acts, e.g. running a component, sending data to any other component and so on. Nevertheless, they also refer acts to deliver tasks between components, e.g. requesting a task to a component, informing an action to other component and so on. The language defines the form and meaning of messages, i.e., the communication act and content of messages. Communication act is an action that a component can accomplish to interact with other ones. Content of message refers what a communication act applies to.

The following example can clarify this: the sentence "*supervisor supplies to actuator the action of open the input valve to 70-90%*" describes the situation "*supervisor*" sends the message "*supplying the action of open the input valve to 70-90%*" to "*actuator*". Thus, "*supplying*" is the communication act and "*the action of open the input valve to 70-90%*" is the content of the message from "*supervisor*" to "*actuator*".

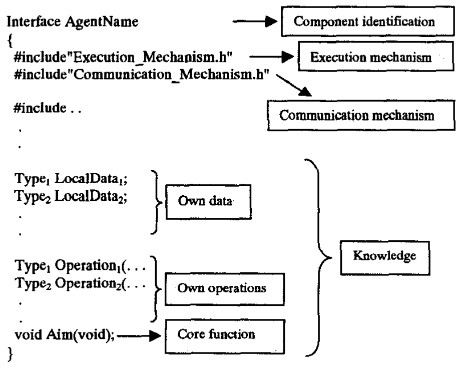


Fig. 5 Partial representation of the software pattern for components with agent entities

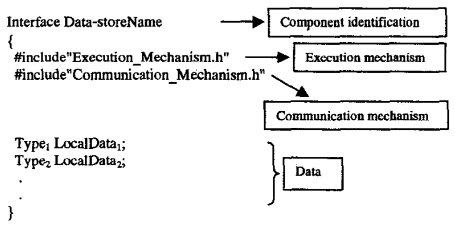


Fig. 6 Partial representation of the software pattern for agent-based data-stores

The communication language defines a set of communication acts, each of which is given a specific meaning. In terms of determination of the meaning of messages, we distinguish three kinds of communication acts so-called *execution acts*, *interaction acts* and *response acts.* Execution acts are those that permit to control the execution of components, e.g. running a component. Interaction acts are those that permit interactions among components, e.g. requesting a task to a component. Response acts are those in response to some other act, e.g. indicating communication faulty. They are:

Execution acts:

- Go: to run a component.
- Ungo: to stop a component.

Interaction acts:

- Put: to put data into a perception base/fact base.
- Get: to get data into a perception base/fact base.
- Supply: to supply data to a component.
- Require: to request to an assistant that accomplishes a specified task. Response is in waiting.
- Request: to request to an assistant that accomplishes a specified task. Response is not in waiting.
- Send: to send result of a requested task.
- Respond: to inform that a requested task was not accomplished.
- Cancel: to cancel some task previously requested.
- Inform: to inform about communication states to co-ordinator/other component.

Response acts:

- OK: to indicate that the communication with other component was successfully accomplished.
- TIME_OUT: to indicate that the communication with other component was not successfully accomplished because the time to establish the communication was exceeded.
- NOT_FOUND: to indicate that the communication with other component was not successfully accomplished because that component is not running.
- NOT_KNOWN: to indicate that a specified data/task was not known/understood.

2) Communication Protocol: It is basically the definition/implementation of the communication acts as common communication patterns. So, the communication acts have syntax defined in the following ways:

For the execution acts the syntax is:
<execution act>(<component name>);

For example, in the situation "*co-ordinator runs assistant*", the instruction is: *Go(assistant);*. "*Go*" is the act what "*co-ordinator*" sends to "*assistant*" so that it starts activities.

For the interaction acts the syntax is:
<interaction act>(<component name>, <content>);
<content> is the content of the message.

For example, in the situation "*supervisor cancels the multiplication solicited to assistant*" the instruction is: *cancel(assistant, multiplication);*. "*cancel*" is the act what "*supervisor*" sends to "*assistant*"; "*multiplication*" is the content of the message, which the act is applied on.

For the response acts the syntax is: <response act>

For example, in the situation "*supervisor cancels the multiplication solicited to assistant*", thereby "*assistant*" confirms "*OK*" to indicate what the solicitation was successfully accomplished. "*OK*" is the response, which "*assistant*" returns to "*supervisor*".

3) Communication Broker: It is the middleware responsible of deliver messages among components, i.e., it is the communication channel (see the Fig. 7). It transmits messages between components, which inhabit into applications where they would execute (independently of others). Applications could be distributed into different machines on network.

IV. EXAMPLE

In order to clarify the proposed approach, we consider a hot-and-cold laboratory plant[5] connected to a software application used as control-panel. An I-SS supervises the plant. This is an academic example accomplished to verify desirable features in I-SSs such as interaction of partial supervisors to reach global supervision, distribution of data/knowledge, sharing tasks/results and so on.

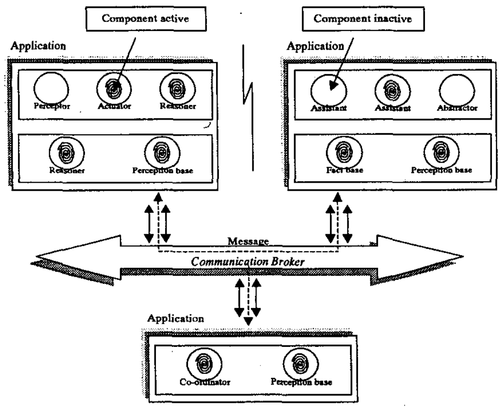


Fig. 7 Schematic of communications among components

The hot-and-cold plant is a simple closed water circuit. It consists of a primary circuit, a secondary circuit and a heat-exchanger (see the Fig. 8). In the primary circuit, multiple boilers warm up the water. Multiple pumps force the warm water to the heat-exchanger. In the heat-exchanger, the water into the primary circuit warms up the water into the secondary circuit. In the secondary circuit, a pump forces the warm water to a refrigerator, which cools down it. Then, the water passes again through the heat-exchanger. Multiple open/close valves and a three-way valve are both on the primary circuit and on the secondary circuit. Open/close valves permit/obstruct to pass the water, e.g. the valve v_1, in the Fig. 8, permits/obstructs the water to the heat-exchanger. The three-way valve on the primary circuit permits/obstructs the water from the pumps/heat-exchanger to the boilers. The three-way valve on the secondary circuit permits/obstructs the water from the pump to the heat-exchanger. Closing or opening valves would cause malfunctions on the plant behaviour.

Control-panel permits to start/stop the boilers and pumps and to manipulate the valves on the plant, and to get temperature and pressure measurements from it. Sensors and actuators (devices) are linked (via a data-acquisition card) to this application. Sensors acquire water temperatures and pressures and boiler temperatures and pressures, e.g. from the points t_1, t_2, t_3 and t_4 and boilers b_1 and b_2 in the Fig. 8. Actuators execute set point changes on the boilers, pumps and valves, e.g. on the valves v_1, v_2, v_3, v_4 and v_5 in the Fig. 8.

Behaviour situations in the primary and secondary circuits determine the global plant behaviour. The following are behaviour situations:

In the primary circuit:

- "*When the temperature in the point t_1 is lower than 52ºC, close the valve v_1 and open the three-way valve*".
- "*When the temperature in the point t_2 is upper than 90ºC, stop the boilers*". We consider that the water temperature must not be upper than 90ºC (in any point of the circuit).
- "*When the temperature in the boiler b_2 is upper than 90ºC, stop it*". We consider that the boiler temperatures must not be upper than 90ºC.

In the secondary circuit:

- "*When the temperature in the point t_3 is cool, close the valve v_4 and stop the pump*".
- "*When the temperature in the point t_4 is cool, open the valve v_5 to 50-75%*". In this situation, we consider that the valve v_5 has 4 positions: 0-25%, 25-50%, 25-75% and 75-100%.

[5] The hot-and-cold plant is a didactic industrial process located in the Universitat de Girona.

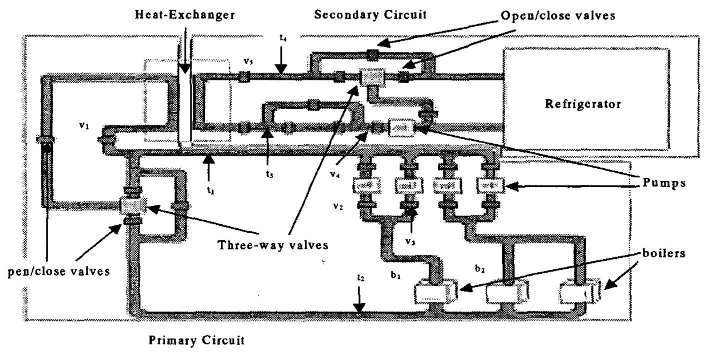


Fig. 8 Schematic of the hot-and-cold plant

Two sub-PBMs that capture separately the behaviours of the primary and secondary circuits were obtained. Analytic/heuristic knowledge were used and integrated to interpret the behaviour situations.

The following example can clarify this: the sentence "*when the temperature in the point t_4 is cool, open the valve v_5 to 50-75%*" describes the situation "*the water temperature experiments the behaviour of cool*" that is happening in the point t_4, thereby is needed the action "*open the valve v_5 to 50-75%*" in order to pass the water to the heat-exchanger to warm it. In this description the symbol "*cool*" is an ambiguous interpretation of the behaviour of the variable "*temperature*" and the symbol "*50-75%*" is an imprecise value of the measurement to open the valve v_5. So, problem-solving techniques based on heuristic knowledge are used to qualitatively interpret the variable measurements (e.g. too cool, cool, warm and too warm) and the valve positions (e.g. 1, 2, 3 and 4, which indicate the opening of 0-25%, 25-50%, 25-75% and 75-100% respectively). Qualitative interpretation permits to identify the fault (if measured temperature is cool). Associations among symptoms and kinds based on if-then rules allow the diagnosis of this fault, e.g. if *temperature in the point t_4* is *cool* then *the water is cool.* Associations among faults and actions based on if-then rules allow to decide actions, e.g. if *water temperature in the point t_4* is *cool* then *open the valve v_5 to the position 3.*

Different components were developed on the two obtained sub-PBMs. The components are:

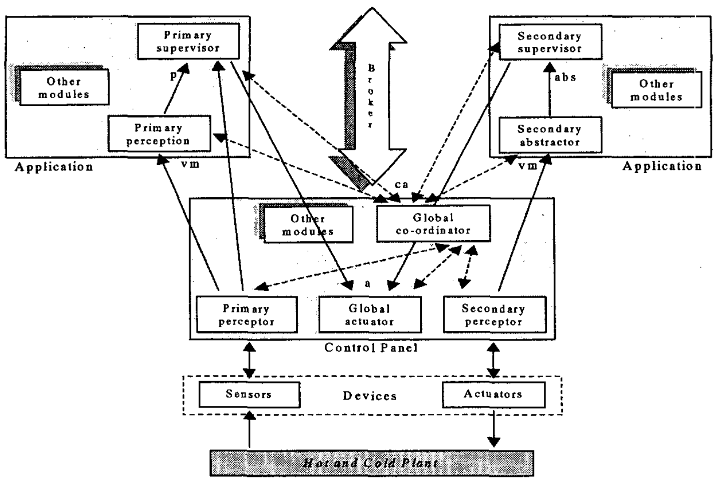
- *Primary perceptor*: it is in charge of perceiving measurements of water temperature/pressure and boiler temperature/pressure from the primary circuit.
- *Primary perception*: it constitutes a store of the measurements supplied by primary perceptor.
- *Primary supervisor*: it supervises the primary circuit. It reasons on perceived measurements (supplied by primary perceptor and stored into primary perception) to detect and to diagnose possible faults and to propose actions to cope with them (to global actuator).
- *Secondary perceptor*: it is in charge of perceiving measurements of water temperature from the secondary circuit.
- *Secondary abstractor*: it obtains qualitative values from the measurements supplied by secondary perceptor. It supplies secondary supervisor with that information.
- *Secondary supervisor*: it supervises the secondary circuit. It reasons on qualitative values (supplied by secondary abstractor) to detect and to diagnose possible faults and to propose actions to cope with them (to global actuator).
- *Global actuator*: it executes set point values on boilers, pumps and valves supplied by primary supervisor and secondary supervisor.

A component so-called *global co-ordinator* was developed to control/co-ordinate the executions/interactions among the before mentioned components. It also constitutes the interface between the control-panel and those components. Interactions among the components and with the control-panel and plant are depicted in the Fig. 9.

Primary perceptor, secondary perceptor, global actuator and global co-ordinator inhabit into the control-panel application where they execute. This is due to the devices are linked to this application and it communicates with global co-ordinator when the plant supervision is needed. Primary perception and primary supervisor, and secondary abstractor and secondary supervisor inhabit into two different applications where they execute. This is due to the differentiation of supervisors.

The followings are obtained results:

- The treatment of the information with two sub-PBMs, which capture separately the behaviours of the primary and secondary circuits, allowed an easier analysis and interpretation of the global hot-and-cold plant behaviour.



vm: variable measurements
abs: abstractions
p: (updates, historical) variable measurements
a: actions
ca: communication acts

Fig. 9 Interactions among the I-SS components, control-panel and hot-and-cold plant

- The integration of different software techniques according to the interpretation of the sub-PBMs allowed an easier manipulation of the information.
- The flexibility of the components as self-defined modules allowed an easier manipulation of them, i.e., an easier adaptation, execution, co-ordination and so on.
- The appropriated processing of the communication acts during the interactions among components allowed an easier control of the inter-operations among them, and subsequently with the control-panel and plant.

V. CONCLUSIONS

The SS complexity is coped with specialised components tuned to solve simple tasks that should operate under co-ordination, which are based on concepts of software agents. This approach is called *"Integrated Supervision Systems, I-SS"*. It proposes multiple supervisors should supervise multiple sub-processes of a process, whose interactions should permit to supervise the global process. According to this goal, multiple sub-PBMs from the process behaviour might be obtained and integrated. Subsequently, multiple components should be developed on those sub-models. Sub-PBMs should capture the sub-process behaviours whose interactions should capture the global behaviour; components should accomplish activities to achieve the supervision goals. Analytic/heuristic knowledge based techniques/methods are used and combined for the analysis and interpretation of situations (more specifically of variables, parameters and relations that describe them).

Different components with predetermined charges and models for the interpretation of situations (both of process behaviours and of interactions among components) were defined and presented. They determine the approach I-SS. A software platform developed to apply the approach was also presented. It constitutes the computational support to develop and to integrate components.

An example that refers an I-SS to supervise a hot-and-cold laboratory plant connected to a software application used as control-panel allowed clarifying the approach. The I-SS consists of different components that were developed on two sub-PBMs obtained from the behaviours of two sub-circuits of the plant.

With the proposed approach a set of desirable features we gathered up. They are:

- Sub-process behaviours are easier to analyse and interpret than the behaviour of huge/complex processes.
- Components are easier to understand, to build and to modify than huge applications. Also, they are easier to adapt to process configurations.
- None of components of an I-SS should have global view of the solutions. They should share information (perceptions, knowledge, results and so on) to reach global solutions.
- If two tasks are functionally similar, one same component could achieve them. Also, once a set of components have been constructed for an I-SS, it should be possible to construct new ones that use these components.
- If a process changes, the modifications on the I-SS must be done only on the components where the changes are involved. Also, it should be possible to replace and/or to add components to modify the I-SS structure according to process configurations.

- Components could be located in different logical/physical points according to the nature of the process with which they interact. That is, they would inhabit into applications where they would execute (independently of others), which could be distributed into different machines on network.

VI. ACKNOWLEDGMENTS

This work is partially funded both by projects CICYT DPI2000-0658 "Diseño de Agentes Físicos Dinámicos, Aplicaciones Futuras" and by projects TAP98-0955-C03-02 "Diseño de Agentes Físicos (DAFNE) / Physical Agents Design".

Special thanks to the LCE (UNAB, Colombia) because it framed the use of I-SS tools with the institutional project UNAB "Componentes Interactivos para Supervisión de Procesos".

VII. REFERENCES

[1] P. Frank and S. Köppen, "New Developments using AI in fault Diagnosis", *IFAC/IMACS International Workshop*, December 1995.

[2] R. Isermann and P. Ballé, "Trends in the Application of Model-based Fault Detection and Diagnosis of Technical Processes", *in Proceedings of the 13th Triennial World Congress*, 1996.

[3] M. Wooldridge and N. Jennings, "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review*, vol. 10, no. 2, 1995.

[4] S. Russell and P. Norvig, *Inteligencia Artificial, Un Enfoque Moderno*, Prentice Hall, Mexico, 1996.

[5] T. Laffey, P. Cox, J. Schmidt, S. Kao and J. Read, "Real-time Knowledge-based Systems", *AI Magazine*, Spring 1998.

[6] J. Aguilar, "Knowledge-based Systems for the Supervision of Real-time Control Process", *in Proceedings of the 4th International Symposium on Knowledge Engineering*, May 1990.

[7] R. Sanz, F. Matía, A. Jiménez, R. Galán, A. De Antonio and M. Segarra, "Heterogeneous Software Integration for Intelligent Process Control: The HINT Project", *in Proceedings of the Valencia COSY Workshop*, 1996.

[8] B. Chaib-Draa, "Industrial Applications of Distributed AI", *in Chapter 2: Applications, of Readings in Agents*, Morgan Kaufmann Publishers Inc., USA, 1998.

[9] J. De La Rosa, *Heuristic for Co-operation of Expert Systems, Application to Process Control*, Doctoral Thesis, Universitat Autónoma de Barcelona, Spain, 1994.

[10] B. Moulin and B. Chaub-Draa, "An Overview of Distributed Artificial Intelligence", *in Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, USA, 1996.

[11] N. Jennings, E. Mamdani, J. Corera, I. Laresgoiti, F. Periollat, P. Skarek and L. Zsolt, "Using ARCHON to Develop Real-world DAI applications, Part 1", *IEEE Expert*, December 1996.

[12] R. Sanz, "Methodologies for Complex Control Systems Engineering", *in Proceedings of the COSY Workshop on Integration of Complex Systems*, August 1998.

[13] G. Fiol-Roig and M. Ferrer, *Expert System for Supervision of Real Time Control Process, Research report*, Universitat de les Illes Balears, Spain, 1998.

[14] O. Contreras, *Interactive Software Agents for Expert Process Supervision*, Preliminary Research Work, Departament d'Electrònica, Informàtica i Automàtica, Universitat de Girona, 2000.

[15] A. De Antonio and M. Segarra, *ICa, an Intelligent Control Architecture, Advanced User's Guide*, Autonomous Systems Laboratory, Universidad Politécnica de Madrid, 1998.