ONTOLOGY FOR INTEGRATING HETEROGENEOUS TOOLS FOR SUPERVISION, FAULT DETECTION AND DIAGNOSIS

Beatriz López, Joaquim Meléndez, Silvia Suárez
Universitat de Girona, Capus Montilivi, edifice P4, 17071 Girona, Spain
Email: {blopez, quimmel,sasuarez}@eia.udg.es

Keywords: Distributed control systems, Intelligent fault detection and identification, Industrial expert systems.

Abstract: The Distributed Supervision Systems that have been used extensively for the last fifteen years in the process industry are now evolving towards higher level solutions based on better connections between applications and processes that assure that data flows from the process to manage boards. Knowledge sharing seems to be a key issue in integrating these heterogeneous systems. In this paper we present an ontology as a first step to achieving semantic interoperability. The ontology has been conceived within the context of a complex integration problem, in which heterogeneous toolboxes cooperate to deal with several supervision, fault detection and diagnostic tasks for chemical processes. Regarding the current trends in ontology research, our proposal is consistent with top-level ontologies, as these kinds of ontologies seem to overcome the ontology integration problem. We describe a preliminary version of the ontology. The conceptualisation of control variables, system behaviour, supervision tasks, models and system properties is given. All attributes and relationships between each concept has been deployed. The ontology has been developed using Protete2000.

1 INTRODUCTION

The Distributed Supervision Systems that havebeen used extensively for the last fifteen years in the process industry are now evolving towards higher level solutions based on better connections between applications and process that assure that data flows from process to manage boards. Current requirements of flexibility, traceability and quality mean that all agents (suppliers, factories, vendors, maintenance, etc.) that participate in the final product must communicate with each other continually. Figure 1 shows the basic architecture of a SCADA (Supervisory Control and Data Acquisition) software. It is clearly oriented towards guaranteeing the integration of the process (Instrumentation Communication Interface), operators (HMI), supervisors (SPC/SQC), manager and other enterprise resources (ERP).

Advances in distributed and ubiquitous computing, networking and sensors provide new environments in which it is possible to integrate these supervision techniques (Murray et al, 2003). However, integrating supervision techniques presents the challenge of shifting from traditional supervision systems as processes with single controllers to supervision systems as collections of heterogeneous physical and information systems with complex inter-connections and interactions (Murray et al, 2003; MacFarlane and Bussmann, 2000).

Holonic Multiagent systems seem to be a promising paradigm for managing, modelling and supporting integration (MacFarlane and Bussmann, 2000). They provide a common platform to facilitate the information flow among different heterogeneous systems in such a way that decision support systems can be improved. The burden of information access, extraction and interpretation in the different steps that constitute supervising a plant is automated and presented to the human operator in a holistic and more comprehensive way.

However, using a Holonic Multiagent system first requires the current supervision systems to be encapsulated, and second it requires elaborating a common, shared vocabulary that provides semantic interoperability among the different systems. Our goal here is to develop an ontology that provides semantic interoperability. In particular, we focus on integrating heterogeneous tools for supervision, fault detection and diagnosis (SFDD) within the

context of the CHEM European project (Cauvin, 2002).

This paper is organized as follows: First, in section 2 we explain what holonic multi-agent systems are and in section 3 we make some ontology definitions. In section 4, we provide the details of the SFDD ontology. In section 5 we explain how this ontology can be used to integrate heterogeneous toolboxes and we end with some conclusions in section 6.

2 HOLONIC MULTI-AGENT SYSTEMS

Holonic Multi-Agent research concerns two main communities: holonic manufacturing systems and agent technology. On one hand, a holon is "an autonomous and co-operative building block of a manufacturing system for transforming, transporting, storing physical and information objects" (MacFarlane and Bussmann, 2000). It consists in a control part and an optional physical processing part. A holon can be made up of other holons (MacFarlane and Bussmann, 2000; Giret and Botti, 2004). This concept of holons is clearly an extension of the current SCADA systems (See figure 1) with improved communication and processing capabilities clearly oriented towards decision making.

On the other hand, agents provide autonomy with respect to the system capacity for a given environment (Wooldridge, 2002). Agent Technology, although broadly extended in open applications such as Internet services, has only recently been introduced to the supervision field (see (Bussmann and Schild, 2001) for an example).

Although both approaches, holons and agents, share many basic concepts, research into each area has mainly been developed independently: research into holonic systems has focused on manufacturing systems, and research into Agent Technology has focussed on developing interconnected systems in which data, control, expertise or resources are distributed. Being aware of the common interest, there have been recent efforts to understand whether holons and agents are different or not and to join both communities (Marik et al., 2003)). This has lead to the term Holonic Multiagent Systems being formed, a novel paradigm for managing, modelling and supporting complex systems. This new paradigm provides two main benefits. On one hand, holons provide soundness and robustness, typical characteristics of the system engineering developments. On the other hand, agents facilitate the integration of heterogeneous systems.

Our work is in line with this new approach for integrating heterogeneous supervision, fault detection and diagnosis systems. More than building new SFDD techniques, we focus on integrating them. As a first step, we have participated in developing toolboxes that encapsulate and describe SFDD techniques. Currently, we are dealing with the problem of making interconnecting toolboxes operational. In this challenge, one of the main drawbacks consists in information sharing, and therefore, ontologies play an essential role.

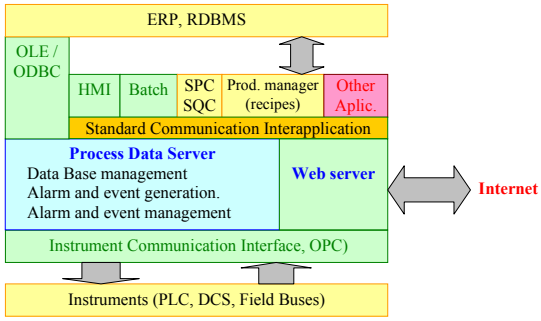

Figure 1: Basic structure of a SCADA system (from (Issermann and Ballé, 1997))

3 ONTOLOGIES

In the past research into ontologies was rather confined to the philosophical sphere. Currently it is widespread in research fields as diverse as knowledge representation, knowledge engineering, qualitative modelling, database design, information systems and database integration, natural language understanding, information retrieval and extraction, object-oriented software development, knowledge management and organization, and agent-based system development (Giunchiglia et al., 2003). Several standardization organisms such as ISO, IEEE, and W3C are now working on this new technological challenge to integrate systems by means of a common vocabulary.

Ontologies can be seen as metadata that explicitly represent semantics of data in a machine processable way (Giunchiglia et al., 2003). By making the link between the information's form and content explicit, ontologies help people and computers to access the information they need. Moreover, ontologies are now recognized as powerful tools that enable

sharing knowledge, and a growing number of applications have benefited from using ontologies as a means of achieving semantic interoperability among heterogeneous, distributed agent systems (Sure and Corcho, 2003). They therefore have a crucial role in integrating supervision techniques.

An ontology defines a common vocabulary for researchers who need to share information in a domain. A domain ontology corresponds to an organized set of domain generic terms that can be used to describe a particular domain by providing machine-interpretable definitions of basic concepts in the domain and the relationships between them (Noy and McGuinness, 2001). An ontology of a specific domain is useful in two aspects: first, to make understanding the process in a specific domain easier; and second, to obtain a standard representation that can be shared and reused in other tools. With the second point it is important to highlight that different tools have been developed by several designers and there is no common vocabulary, so ontologies seem to be an appropriate mechanism for integration.

Recent research work, however, has experimentally proved that ontologies are not enough to guarantee semantic interoperability. In (Correa et al., 2002) four main problems have been detected: 1) reusing ontologies to engineer new ontologies is not straightforward; 2) ontologies do not provide adequate information when sharing inferences; 3) when reasoning under uncertainty, additional semantic links regarding inference are required, and 4) in a large scale system, sharing group knowledge should be appropriately studied.

Regarding the first problem, Guarino observes that ontologies developed from a bottom-up approach based on multiple local ontologies, may not work because they focus on conceptual relations in a specific context (Guarino, 1998). Therefore, there is no guarantee that two systems with the same vocabulary have the same conceptualisation. This is what he calls the ontology integration problem.

In order to deal with this problem, several authors argue in favour of mapping mechanisms between ontologies (Schorlemmer and Kalfoglou, 2003), while others, such as (Guarino, 1998), propose using different kinds of ontologies. Guarino distinguishes between top-level, domain, task and application ontologies, as shown in figure 2. Top-level ontologies provide constraints and building blocks for representing knowledge (Martin and Eklund, 1999). Domain level ontologies describe the vocabulary related to a generic domain. Task ontologies are related to generic tasks or activities. Finally, application ontologies describe concepts depending on a particular domain and task.

In this paper we propose a top-level ontology for distributed supervision systems (supervision, fault detection and diagnosis). Other works are related to supervision but at the application level. In (Bernaras et al., 1996), the authors present an ontology for fault diagnosis in electrical networks. In (Kitamura and Mizoguchi, 1999) the authors provide an ontological analysis at the task level regarding fault processes. Another interesting work on ontologies is in WEDSS, used to integrate rule-based systems, case-based reasoning and classical control techniques for wastewater management (Ceccaroni et al., 2004).

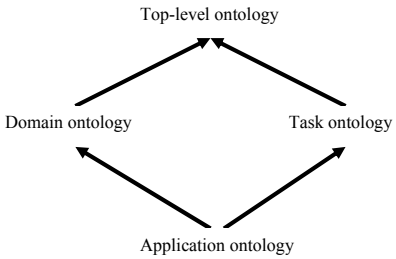

Figure 2: Kinds of ontologies according to (Guarino, 1998)

4 SFDD ONTOLOGY

In order to elaborate an ontology for SFDD tasks, we have used the terms proposed in (Isermann and Ballé, 1997) and (Colomer and Meléndez, 2000) for supervision, fault detection and diagnosis. Therefore, the main terms are organized in variables, system behaviours, supervisory tasks, models and system properties (see Figure 3). Each term is defined in properties and relations, generating a complex network of classes, subclasses, instances and slots. (See figure 4 for a detailed description of the terms). In the following section we describe all the terms.

Regarding the linking between ontologies and the conceptual modelling of the overall system, our ontology is a top-level one as stated above. So, first the particular instantiation of models, system variables and system behaviours will lead to a domain ontology for a given modellization. Second,

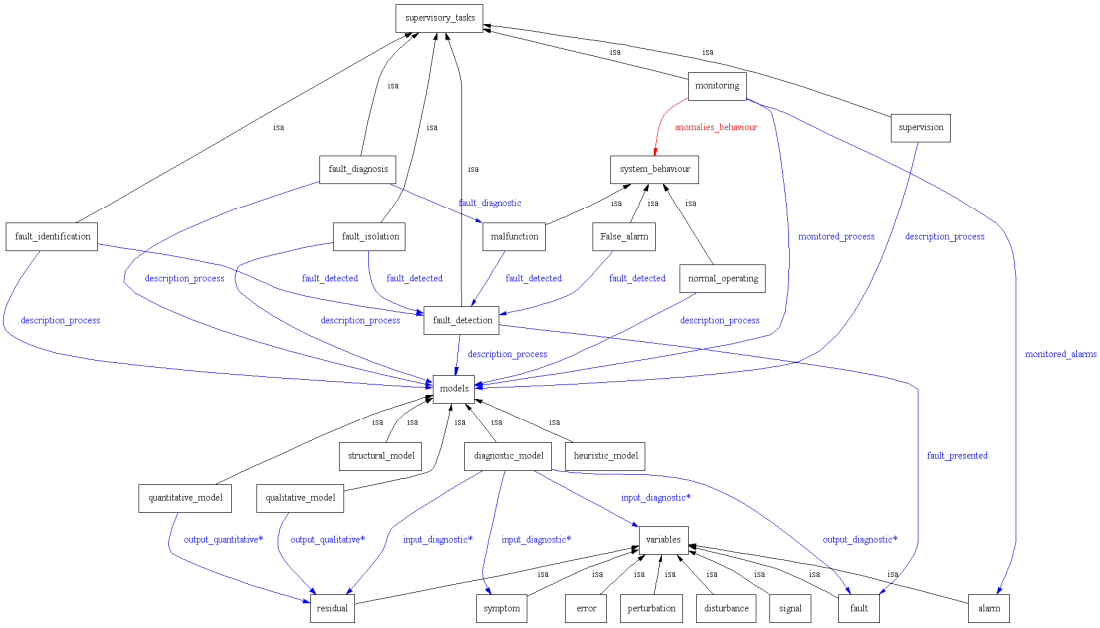

Figure 3: General diagram of the ontology

the refinement of supervisory tasks will provide a task ontology. And finally, the enhancement of system properties will derive in an application ontology.

4.1 Variables

The following variables have been considered for SFDD purposes and characterised as follows:

- Signals are defined as physical measures or perceptible variables that communicate information and messages. Attributes: Units, Range, Typology (quantitative/qualitative, continuous/sampled/discrete events).
- Fault: Unpermitted deviation of at least one characteristic property or parameter of the system in acceptable / usual / standard conditions. It is composed of the following attributes: cause, duration, final_time, typology_of_fault (intermittent/permanent, evolutive/abrupt, additive/parametric), location_fault, Descriptor_fault, Size_fault, starting_time.
- Error: Deviation between a computed variable (typically and output or state variable) and the true, specified or theoretically correct value. It has the following attributes: two inputs (to compute it): correct_value, duration, final_time, measured_computed_value, result, starting_time
- Disturbance: An unknown (and uncontrolled) input acting on a system. The attributes of this subclass are: typology (additive /multiplicative, etc), shape, duration final_time, input_point (if known, related
with the physical system and/or the structural model), effects and starting_time.
- Perturbation: An input acting on a system which results in a temporary departure from the current state. The attributes are: idem_disturbance.
- Residual: Fault indicator, based on deviations between measurements and model-equation-based computations. Particular case of error. It has the following attributes: deviation, duration, final_time, result (fault detection decides about the presence or absence of faults), starting_time, decision_mechanism, Signature.
- Symptom: An observable quantity changes its normal behaviour. The attributes of this subclass are: referred_quantity, duration, final_time, starting_time, shape (trends) and values.

4.2 System Behaviour

System behaviour is an overall description of the system operating conditions. Basic states can be defined:

- Normal Operating: the system behaves according to the specifications.
- Faulty or Malfunction: Intermittent irregularity in fulfilling a system's function. It has the following attributes: periodicity, starting time, final_time, faults (see fault attributes).
- False alarm: The system is operating properly but the supervisory system has detected some misbehaviour.

Fault_diagnosis		
Description_process	Instance	Models
Time_detection	Float	
Diagnostic_location_fault	String	
Diagnostic_size_fault	String	
Fault_diagnostic	Instance	Malfunction
Diagnostic_typology_fault	String	

Figure 4: Description of the fault diagnosis attribute

4.3 Supervisory Tasks

The tasks are the different kinds of operations that must be performed in order to supervise a given system. There are seven main kinds of tasks:

- Fault detection: Detection of faults in a system and the detection time of a fault. This subclass is composed of the following attributes: fault_presented (yes /no, without specifying additional information) and time_detection.
- Fault isolation: Identification of the relevant attributes of the faults present: kind, location and detection time of a fault. This task follows fault detection. The attributes of this subclass are: fault_presented and time_detection (input attributes given by the fault detector), fault attributes (kind_fault, location_fault, fault_time) are presented as the conclusion of this task. This task needs information from specific models of the system (structural model, diagnosis model) in order to perform the task.
- Fault identification: Detection of other relevant attributes of faults: the size and time-variant behaviour of a fault. This task follows fault isolation. It has the following attributes: fault attributes (size_fault and timevariant_behaviour).
- Fault diagnosis: Detection of kind, size, location and the detection time of a fault. This task follows fault detection. Fault group isolation and identification. The attributes of this subclass are: time_detection, fault_presented and diagnostic (kind_fault, location_fault, size_fault and time_detection). It needs structural and/or diagnosis models to perform the task.
- Monitoring: A continuous real time task to determine the conditions of a physical system, by recording information and recognizing and indicating anomalies of the behaviour. The attributes are: monitored_variables, alarms, events, operating_conditions, tunning_parameters (thresholds and similars), anomalies_behaviour and information_recognising.
- Supervision: Monitoring a physical system and taking appropriate actions to keep the system operating in the case of faults. It has the following attributes: diagnostic, actions, decision_system.

4.4 Models

For the purpose of engineering analysis and design, physical systems are usually represented in some mathematical form; this representation is also called the model of the system. The properties of the model reflect the nature of the system, though in many cases the model may just be an approximation of the true system behaviour. In (Isermann and Ballé, 1997), five classes of models are considered (See figure 5):

- Quantitative model: Uses static and dynamic relations between system variables and parameters in order to describe system behaviour in quantitative mathematical terms. The attributes of this subclass are: description, input_quantitative, output_quantitative and parameters.
- Qualitative model: Uses static and dynamic relations between system variables and parameters in order to describe system behaviour in qualitative terms such as causalities or if-then rules. It has the following attributes: description, input_qualitative, output_qualitative and parameters.
- Diagnostic model: A set of static or dynamic relations which link specific input variables (the symptoms) to specific output variables (the faults). The attributes of this subclass are: description, input_diagnostic (symptoms, residuals, and physical variables), output_diagnostic (fault attributes) and parameters.

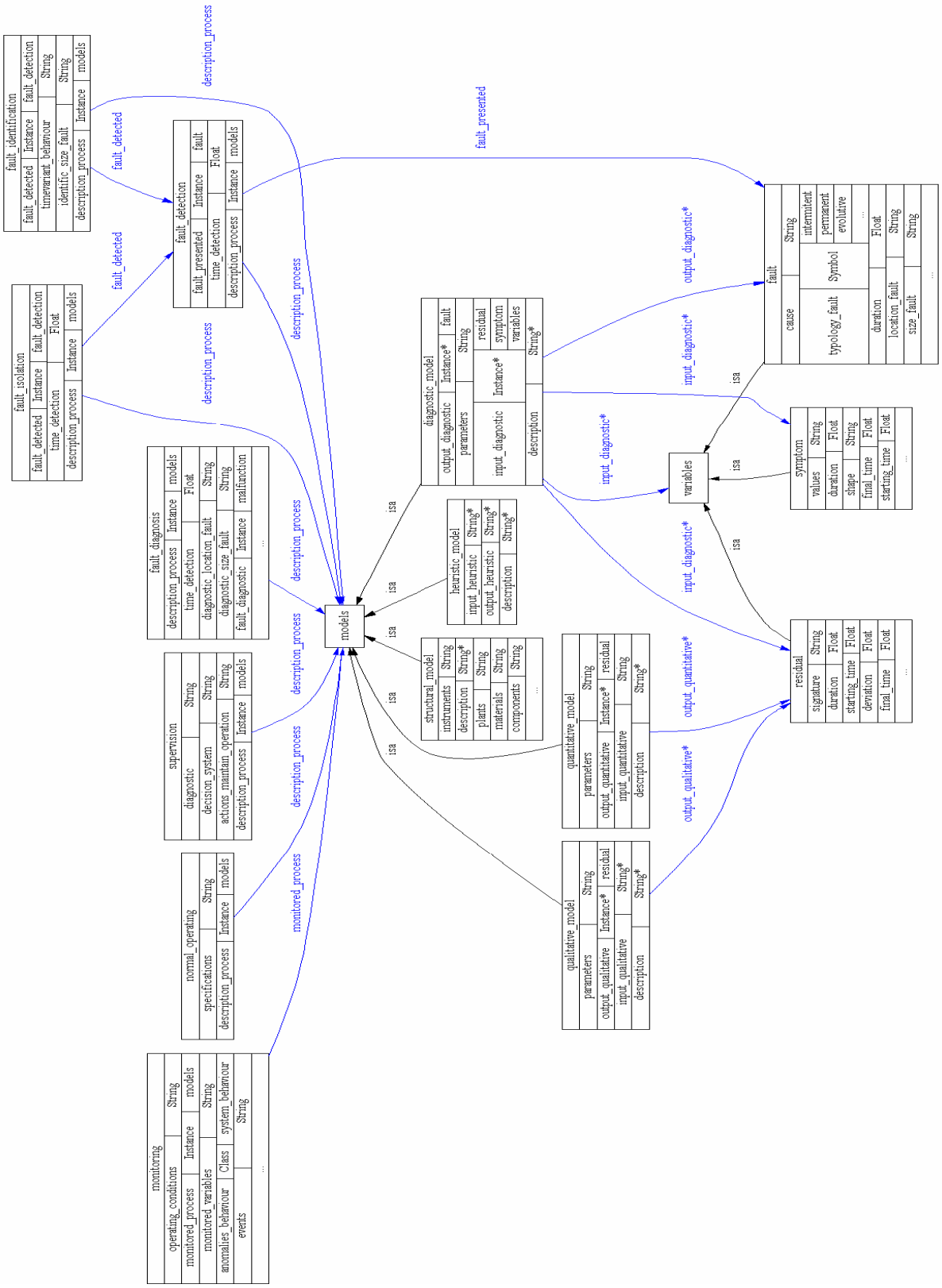

Figure 5: Ontology particular diagram

- Heuristic_model: The attributes of this subclass are: description, input_heuristic and output_heuristic. Commonly used for making diagnoses, working with symptoms and attributes of the diagnostic model.
- Structural model: Definition of the physical interaction between components, materials and energy sources. Attributes: description, components, (sub)systems, instruments, plants, materials.

In a future, we will add a sixth class regarding event-driven models.

4.5 System Properties

System properties relate particular characteristics required for the system. The main properties are: reliability, safety, availability, dependability.

- Reliability: Ability of a system to perform a required function under stated conditions, within a given scope, during a given period of time. Measure: MTBF = Mean Time Between Failure. MTBF = 1\la; la is rate of failure (e.g. failures per year). It has the following attributes: MTBF, period_time and required_function.
- Safety: Ability of a system not to put people, equipment or the environment into danger. It has the following attributes: value_safety.
- Availability: Probability that a system or the equipment will operate satisfactorily and effectively at any point of time measure: MTTR Mean Time To Repair MTTR = 1/μ; μ: rate of repair. The attributes of this class are: MTTR and probability_availability.
- Dependability: A form of availability that has the property of always being available when required. It is the degree to which an item is operable and capable of performing its required function at anyrandomly chosen time during its specified operating time, provided that the item is available at the start of that period (RAM Dictionary). It has the following attributes: degree_dependability and time_dependability.

5 ONTOLOGY FOR SFDD

The ontology described in the previous section has been created using the tool Protegé 2000 (Noy et al., 2003) with the purpose of integrating different supervision, fault detection and diagnosis toolboxes, within the context of the CHEM project (Cauvin, 2002). These toolboxes are the result of encapsulating SFDD techniques which provide a common description and interface for users. Each toolbox has been designed and developed by different teams. The SFDD ontology provides a shared and common vocabulary for the toolboxes with two main benefits: Firstly, the operator handles decision support information holistically. Secondly, the operator is not burdened with different vocabularies and interpretations coming from heterogeneous tools, but can work with a single ontology. Figure 6 shows the differences between the current SFDD information flow and the integration approach using an ontology.

6 CONCLUSIONS

In this paper we present a top-level ontology for sharing knowledge in distributed supervision systems. We provide the basic conceptualisation and implementation with Protege2000. The ontology is presented as a first step towards SFDD toolbox interoperability.

We have two main lines of research for future work. First, to study pruning and factoring mechanisms, like in (Conesa et al, 2003), in order to derive both task and domain level ontologies from our top-level ontology. Second, to effectively integrate SFDD toolboxes into a multi-agent platform (Wooldridge, 2002) by means of the ontology.

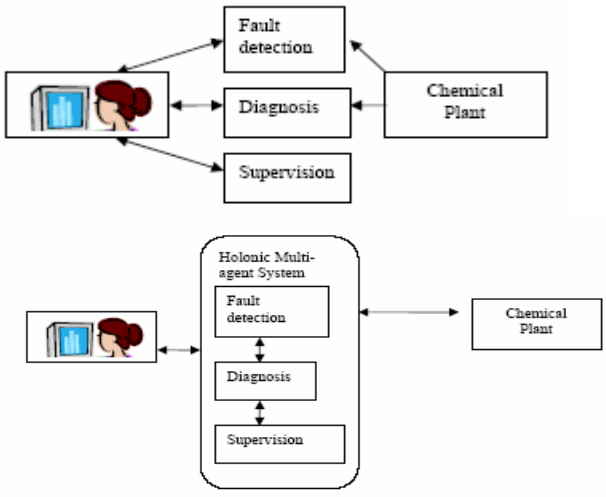

Figure 6: Above: current data flow of heterogeneous SFDD tools. Below: Data flow of supervision, fault detection and diagnosis integration

ACKNOWLEDGMENT

This work has been supported by the Spanish MCYT project DPI2001-2198 and MEC TIN2004-06354-C02-02.

REFERENCES

Bernaras A., Laregoiti I., Bartolomé N. And Corera J., 1996. An Ontology for Fault Diagnosis in Electrical Networks. *Proc. Intelligent Systems Applications to Power Systems, (ISAP '96)*.

Bussmann, S., Schild, K. 2001. An Agent-based Approach to the Control of Flexible Production Systems. *Proc. 8th IEEE Int. Conf. on Emerging Technologies and Factory Automation* (EFTA 2001), pp. 169-174..

Cauvin, S., 2002. CHEM:. Advanced Decision Support System for Chemical/Petrochemical Manufacturing Processes. *Monet Newsletter*, issue 2.

Ceccaroni, L., Cortés, U., Sánchez-Marré, M., 2004. OntoWEDSS: augmenting environmental decision-support systems with ontologies. *Environmental Modeling & Software* 19 785-797.

Colomer J, Meléndez J., 2000. "Sistemas de Supervisión". *Cuadernos CEA-IFAC*. Ed. CETISA.

Conesa Jordi, De Palol Xavier, Olivé Antoni. 2003. Building Conceptual Schemas by Refining General Ontologies. In V.M.W.R.O. Stepenkove, editor, *DEXA'03* , volume 2736 of LNCS, pp. 693-702, Springer. .

Correa da Silva F. S., Vanconcelos W., Robertson D.S., Brilhante V., de Melo A.C., Finger M., Agustí J., 2002. On the insufficiency of Ontologies: Problems in Knowledge Sharing and Alternative Solutions. *Knowledge-based systems*, 15(3): 147-167.

Giret, A., V. Botti, V. 2004. Holons and Agents. *Journal of Intelligent Manufacturing*. 15(35):645-659.

Giunchiglia, F., Gomez-Perez, A., Stuckenschmidt, H., Pease, A., Sure, Y., and Willmott S. (eds), 2003. 1st International Workshop on Ontologies and Distributed Systems (ODS2003).

Guarino N., 1998. Formal Ontology and Information Systems. *Proceedings of FOIS'98*, IOS Press, pp. 3-15.

Isermann and Ballé, 1997. "Trends in the application of model based fault detection and diagnosis of technical processes", *Control Engineering Practice*, vol. 5, n° 5, pp. 709-719

Kitamura Y. and Mizoguchi R. 1999. An Ontological Analysis of Fault Process and Category of Faults. *Proc. Of Tenth International Workshop on Principles of Diagnosis (DX-99*), pp.118-128..

MacFarlane, D., Bussmann, S., 2000. Developments in holonic production planning and control. *Int. Journal of Production Planning and Control*, 11 (6): 522-536.

Marik, V., McFarlane, D., Valckenaers, P. (Eds.) 2003: 1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems, Springer, LNCS, Vol. 2744.

Martin P., Eklund P., 1999. Embedding Knowledge in Web Documents. *Computer Networks*. vol 33. No. 11-16. pp. 1403-1419.

Murray, R.M., Aström, K.J., Boyd, S.P., Brockett, R.W., Stein, G. 2003. Future Directions in Control in an Information-Rich World. A summary of the report of the Panel on Future Directions in Control, Dynamics, and Systems. *IEEE Control Systems Magazine*, April 2003, pp. 20-33.

Noy N.F. and McGuinness D.L, 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05..

Noy NF, Crubezy M, Fergerson RW et al. , 2003. Protégé-2000: An Open-source Ontology-development and Knowledge-acquisition Environment. *Proc AMIA Symp.*

Schorlemmer, M., Kalfoglou, Y. 2003. Using Information-Flow Theory to Enable Semantic Interoperability. In: *Artificial Intelligence Research and Development.* I. Aguiló et al. (Eds), IOS Press, 421-431.

Sure Y., Corcho O. (eds), 2003. 2nd International Workshop on Evaluation of Ontology based Tools (EON2003), 2003.

Wooldridge M., 2002. An Introduction to Multiagent Systems. John Wiley & Sons.