Computers and Chemical Engineering 59 (2013) 164–177

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

A knowledge-driven approach for process supervision in chemical plants

Estanislao Musulin[a,*], Fernando Roda[a], Marta Basualdo[a,b]

[a] Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas, S2000BTP Rosario, Argentina
[b] Universidad Tecnológica Nacional-FRRo, Zeballos 1341, S2000BTP Rosario, Argentina

ARTICLE INFO

Article history:
Received 9 August 2012
Received in revised form 22 May 2013
Accepted 3 June 2013
Available online 25 June 2013

Keywords:
Process supervision
Description Logic
Ontology
Alarm management
Tennessee Eastman process.

ABSTRACT

In this work, an ontology-based framework for process supervision in chemical plants is presented. A conceptualization of equipment, control systems and hazards has been developed. This conceptual model includes the semantic of each modeled term in order to obtain a heavyweight ontology. The ontology has been formalized using Description Logic (DL) (Krötzsch et al., 2012). A knowledge-driven approach has been adopted in order to demonstrate how DL reasoning could be used to support process supervision, detecting and diagnosing faults, without the help of external agents. In the proposed approach, a DL reasoner adds implicit facts to the ontology through forward chaining reasoning, from the current measurements to the characterization of hazards. Additionally, the system is able to check knowledge consistency and formally explain the obtained results. The system functionality has been illustrated in the Tennessee Eastman process.benchmark.

1. Introduction

1.1. Process supervision background

Process supervision and fault diagnosis deals with the detection and isolation of abnormal events. It consists of interpreting the current state of the plant from sensor readings and process knowledge. Fault diagnosis is of crucial importance in terms of safety and also of economics, because of the influence of abnormal events in yield and quality of products.

Usually, process operators with the support of control alarms (i.e. simple univariate monitoring alarms), are who manage the abnormal situations. The complexity of modern processes makes difficult the prediction of abnormal events since many different things can go wrong in many different ways. However, control alarms do not give information about root causes of faults, they only inform about some particular deviation. As a consequence, in the last years, intelligent, real-time supervision and fault diagnosis systems arise as the necessary tools to help operators in dealing with abnormal situations. Such a system has to work in real time, detecting abnormal events as soon as possible and informing the operator about the deviations, the root causes and suggesting solutions. Therefore, appropriate actions can be taken in real time in order to avoid faults propagations and tending to reduce possible consequences of abnormal events that may occur.

Many process supervision methods have been reported in literature. These methods can be roughly classified in three sets: quantitative model based methods, qualitative model based methods and historical based methods. Quantitative model based methods use quantitative techniques such as bond graphs models (Ould-Bouamama, El Harabi, Abdelkrim, & Ben Gayed, 2012), observers (Patton & Chen, 1997), and Kalman filters (Bhagwat, Srinivasan, & Krishnaswamy, 2003; Villez, Srinivasan, Rengaswamy, Narasimhan, & Venkatasubramanian, 2011) all of them based on deep process knowledge. Qualitative model based diagnosis has been proposed by Venkatasubramanian and Rich (1988) and Ram Maurya, Rengaswamy, and Venkatasubramanian (2004) using digraph casual models. Historical based methods where presented by Sundarraman and Srinivasan (2003), Maurya, Paritosh, Rengaswamy, and Venkatasubramanian (2010), and Villez, Rosén, Anctil, Duchesne, and Vanrolleghem (2013) using trend analysis and by Moore and Kramer (1986), Muthuswamy and Srinivasan (2003), and Qian, Li, Jiang, and Wen (2003) using expert systems for multivariate inference. Statistical methods (MacGregor & Cinar, 2012; Yu, 2012; Zhang, Ge, Song, & Fu, 2011) and neural networks (Srinivasan, Wang, Ho, & Lim, 2005) are examples of quantitative historical based methods techniques. Additionally, several methods where combined in order to find the proper synergies to amplify the strengths while avoiding the shortcomings intrinsic to each methodology, such as the lack of transparency, extensibility, novel fault detection, system

* Corresponding author. Tel.: +54 341 4237248x331.
E-mail addresses: musulin@cifasis-conicet.gov.ar, estanislao.musulin@gmail.com (E. Musulin).

0098-1354/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compchemeng.2013.06.009

adaptation and robustness (Ghosh, Ng, & Srinivasan, 2011; Musulin, Yelamos, & Puigjaner, 2006; Wang, Zhao, & Shang, 2012). It is not intended to give a complete bibliography of this vast topic, for a more complete review, the reader is referred to the excellent articles of Venkatasubramanian, Rengaswamy, and Kavuri (2003), Venkatasubramanian, Rengaswamy, Kavuri, and Yin (2003), and Venkatasubramanian, Rengaswamy, Yin, and Kavuri (2003) and Maurya, Rengaswamy, and Venkatasubramanian (2007).

Fault detection and diagnosis cannot improve safety and reliability for themselves. A fault evaluation must be performed to classify faults into different hazard classes. Hazards are undesirable system conditions with the potential to cause or to contribute to a damage or accident (Leveson, 1995). Since no action can be made to avoid, or reduce, the effects of unidentified hazards, a process hazard analysis must be performed in advance to the fault diagnosis task. HAZOP (Gillett, 1997) is the most widespread inductive technique to identify hazards and to evaluate possible scenarios leading to unwanted consequences. Since HAZOP is a time consuming and labor-intensive activity many researches have attempted to develop expert systems to automate HAZOP analysis.

One of the first attempts to automate HAZOP was the pioneer work of Parmar and Lees (1987a, 1987b), they presented a rule based algorithm to model and propagate faults for hazards identification. Later on, in a series of works (Vaidhyanathan & Venkatasubramanian, 1995; Venkatasubramanian & Vaidhyanathan, 1994) Venkatasubramanian and his colleagues developed and evolved a model-based framework and an expert system called HAZOPExpert using G2 Gensym system. In Vaidhyanathan and Venkatasubramanian (1996) the authors proposed a semi-quantitative reasoning methodology to rank and filter HAZOPExpert results deemed as minor or unrealistic by the human experts. Zhao, Bhushan, and Venkatasubramanian (2005a, 2005b) introduced a high quality full-scale software system (PHASuite) for automated HAZOP analysis. PHASuite was based on a knowledge engineer framework, including a reasoning engine based on Petri Nets. Finally, Zhao, Cui, Zhao, Qiu, and Chen (2009) integrated six existent ontologies in a Case-Based Reasoning system that automatically generates HAZOP analysis. Intelligent HAZOP analysis approaches has been extensively reviewed in Venkatasubramanian, Zhao, and Viswanathan (2000), Zhao et al. (2005a), and Dunjó, Fthenakis, Vílchez, and Arnaldos (2010).

1.2. Ontology-based modeling in process engineering

One important limitation to the generalized adoption of advanced process supervision tools is the necessity of highly customized solutions. In each process, new models have to be developed and a huge amount of data must be analyzed, resulting in an intensive task in terms of time, effort and money. In order to develop more generalized and transparent solutions, the interest in reusable knowledge models arise.

Usually knowledge-based supervision methods (Moore & Kramer, 1986; Musulin et al., 2006; Muthuswamy & Srinivasan, 2003; Qian et al., 2003; Ruiz, Nougués, & Puigjaner, 2001) employ a high number of rules to detect deviations and to classify them into the corresponding fault classes. These methods typically adopt a rule-based knowledge representation scheme or a programming-logic framework. In plant-wide implementations, these rules are distributed all over the information system with little or no logic linkage between them. Hence any modification on the rules bases could generate inconsistencies since its semantic impact in the rest of the system remains hidden.

On account of these drawbacks, ontologies attracted the attention in the Process Systems Engineering (PSE) field as a convenient means for knowledge representation (Gernaey & Gani, 2010; Morbach, Wiesner, & Marquardt, 2009; Natarajan, Ghosh, & Srinivasan, 2012; Zhao, Jiang, & Yang, 2012). The term ontology denotes a conceptual data schema that represents the relevant domain entities and their relations. The widely accepted definition of ontology is "a formal, explicit specification of a shared conceptualization" (Gruber, 1993). Ontologies depend on existing information sources, incorporating semantics thus making the stored information richer and meaningful. One of the most important potentialities derived from the implementation of a high-quality formal ontology is reasoning. Once the information has been properly described, a software reasoner can check consistency and infer new facts.

Nevertheless, making a high-quality ontology is not an easy task as several design considerations must be taken into account. In the PSE context, a knowledge model should support the requirements of integrated software applications meanwhile correctly capturing the domain semantics. The last is critical to successfully exploit domain knowledge and also has the greatest difficulties as concepts themselves involve very complex constructions like ambiguities, synonyms, homonyms, vagueness, etc. Moreover, the meaning of concepts is highly dependent on context. When obviated, this situation leads to the development of *pseudo-ontologies* (Morbach et al., 2009); formal representation schemes that do not consider the principles of integration and reuse of the knowledge base (KB). However, the most important differentiating factor is not the reusability of the respective structure, but its semantic richness. From this point of view, ontologies can be classified in two types (Corcho, Fernández-López, & Gómez-Pérez, 2006): *Lightweight ontologies* are mainly concepts taxonomies that include relationships between concepts but do not include axiomatic definitions to formally define the semantics of its terms. Due to their simple internal design they are not considered full-fledged ontologies. On the other hand, *heavyweight ontologies* model the domain semantics in a deeper way by adding restrictions to *lightweight ontologies*. Those restrictions (i.e. axioms and constraints) clarify the intended meaning of the terms involved into the ontology.

Consequently, literature presents two distinct approaches to employ ontologies in model design (Villa, Athanasiadis, & Rizzoli, 2009). In the *semantic mediation* approach, concepts from ontologies supplement conventional data and models to facilitate information integration and reuse. In the *knowledge-driven* approach, datasets and models are directly represented as instances of ontologies and embody a statement of the system conceptualization, enabling machine reasoning about the system structure that can lead to more sophisticated applications.

Nevertheless, in PSE domain, most of the proposed ontology-based frameworks involve *pseudo-ontologies* or *lightweight ontologies* sometimes as mere technological update of previous works. Moreover many of these knowledge models (Vaidhyanathan & Venkatasubramanian, 1995; Venkatasubramanian & Vaidhyanathan, 1994; Zhao et al., 2005a, 2005b) were designed following a *semantic mediation* or they were not formally implemented.

One of the most remarkable works in PSE domain is OntoCAPE; a large *heavyweight ontology* reported by Marquardt, Morbach, Wiesner, and Yang (2010). OntoCAPE has been developed in a multi-layer architecture with conceptualization of mathematical models, chemical processes, equipment configurations and control systems. It is used for annotation of electronic documents and data storages in order to obtain a consistent representation of these heterogeneous information sources. Batres and Naka (2000) and Batres, Aoyama, and Naka (2002) have presented a multi-dimensional formalism (MDF) that contains four ontologies for modeling the structure, materials and behavior of chemicals plants. Due to an early collaboration between both research teams, MDF and OntoCAPE share some representation schemes.

Some authors have extended OntoCAPE in order to support several applications: Wiesner, Morbach, and Marquardt (2011) present a prototypical ontology-based software tool for the integration and consolidation of distributed design data in chemical process engineering. The authors deals with some efficiency shortcoming on the reasoning process; so their implementation requires powerful CPU clusters to run acceptably fast. Recently Natarajan et al. (2012) presented an extension of OntoCAPE, called OntoSafe, with the necessary information to evaluate process states and conditions against potential faults. OntoSafe adds to OntoCAPE a partial model with a few new concepts to support a multi agent system (ENCORE). However, the reasoning capabilities remain in ENCORE which is responsible for the supervision tasks.

In this work, a *knowledge-driven* approach to on-line process supervision and diagnosis is presented. Supervision tasks are performed by reasoning on ontology-based knowledge models. The proposed KBS captures process events and identifies possible deviations. Additionally, a HAZOP conceptualization has been developed to support Hazard management. A prototype has been implemented using semantic web technologies following the W3C standards. With a Description Logic (Krötzsch, Simancik, & Horrocks, 2012) formalization of a *heavyweight ontology* it is also intended to replace the use of horn-like rules by the proper axioms of each supervision class, and use a semantic reasoner to automatically detect and classify process faults.

The paper is organized as follows. The Ontology and its constitutive modules are explained in Section 2. Knowledge exploitation features are illustrated in Section 3 on the Tennessee Eastman process. Finally, some concluding remarks are given in Section 4.

2. The knowledge model

In this paper, it is argued that the domain ontology should be developed not only in the pursuit of a general and reusable knowledge representation of the domain concepts. In addition, it must be prepared to support the process supervision tasks in a specific plant with minimal implementation/adaptation effort. These goals involve two ontological design principles (Corcho et al., 2006): *usability* (i.e. usefulness for a specific task) and *reusability* (i.e. adaptability to different application context).

Reusability has been achieved by a *heavyweight ontology* designed with a careful election of terms and taxonomies following international standards and recognized best practices. To attain usability, the KB was populated with all the declarative and procedural knowledge to perform the supervision tasks without the needing of external agents intervention or domain-specific reasoning techniques. As a consequence, any standard DL reasoner (e.g. Pellet, FaCT++, etc.) could be used without negative impact, enabling the use of new reasoners, when available, that could increase inference speed and/or quality.

This approach has several implications in software design, maintenance and upgrade because the knowledge (specially the domain logic) remains separated from the programs code. For instance, important changes in the control and supervision politics can be implemented in limited times, immediately transferring the new business logic to the whole plant information system, improving consistency and applications portability. Hence, the impact that changes in the business logic cause across the distributed PSE applications can be controlled.

Although others approaches, like the based on OntoCAPE, has reached good reusability properties, they require a significant implementation effort as lot of concepts must be extended and instantiated, many of which may not be necessary for the application domain. In this sense, we consider that these multipurpose frameworks are weak attempting to minimize ontological commitment (other important Ontology principle).

As it was stated above, we did not neglect reusability, but give a mayor importance to usability including only the necessary concepts to implement supervision methods and to manage risk situations. This also leads to better performance in reasoning tasks, enabling on-line knowledge processing.

2.1. Ontology overview and implementation details

In this work, semantic web technologies (Hebeler, Fisher, Blace, & Perez-Lopez, 2009) are used to acquire, deduce and manage process systems knowledge. The KB uses a formal ontology that gives meaning to the existing data by the establishment of concepts, relations and axioms. This framework provides a dynamic scheme because, as applications add new information, the concepts can evolve to become more precise and useful. Additionally, new instances could be added without a complete understatement of its relations, and they do not need to be bound to any class in particular. This provides the needed flexibility to capture and manage incomplete information, enabling fast adaptations and implementations.

The formal ontology for process supervision has been developed using OWL2 (Web Ontology Language), which is based on the SROIQ Description Logic. This enhances the semantic expressivity respect to OWL1 (SHOIN), allowing richer reasoning deductions and a superior consistency checking. Protégé (Stanford Center for Biomedical Informatics Research, 2012) was used as construction tool to assist the design. The entire ontology comprises 133 classes, 77 properties and 279 axioms; the number of individual instances depends on the case under evaluation. The KB also includes Horn-like rules implemented in SWRL (Semantic Web Rule Language). Ontology elements are organized in three interconnected modules: Equipment, Control and Supervision.

The information contained in the ontology is analyzed by a reasoner that checks consistency and evaluates the process state online when the class or instances are updated.

In this work, the reduction of reasoning times has been an important design requirement as real time process supervision requires rapid response. Therefore, the use of axioms that directly affect the system performance where limited in favor of alternative schemes with the same semantic capability.

2.2. Equipment Module

The Equipment Module contains the concepts associated to the plant physical elements. It is designed as a hierarchical taxonomy based on the ISA-95 (ISA, 2000). ISA-95 defines a hierarchy from the enterprise to the unit/work-cell level but process control is not considered.

For batch processes, the ISA-88 (ISA, 1995) standard allows a more detailed specification from the procedural control point of view. In this context, it is possible (to some extent) to merge both standards to obtain a more complete specification (see Fig. 1) (Brandl, Broersen, Chappell, Charpentier, & Craig, 2007). However, the procedural control model used in batch processes cannot be directly extended to continuous processes. In batch processes, procedures and control equipment are closely tied and unit interactions are limited. This is not the case in continuous processes, where control systems are in general distributed all along the plant.

In this ontology, the standards ISA-95/ISA-88 have been followed until the equipment level, while control system concepts have been specified in a different module (see Section 2.3). Fig. 2 shows fragment of the Equipment class hierarchy, which is designed as a mereology where concepts are linked by aggregation

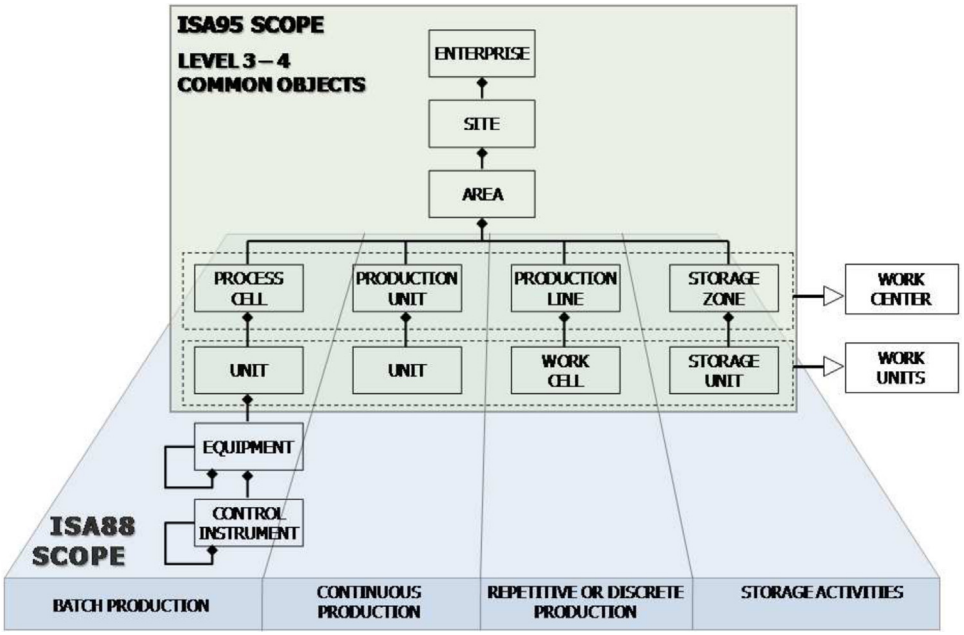


Fig. 1. ISA-88 Part 1 physical model overlaid onto ISA-95 Part 3 expanded equipment.

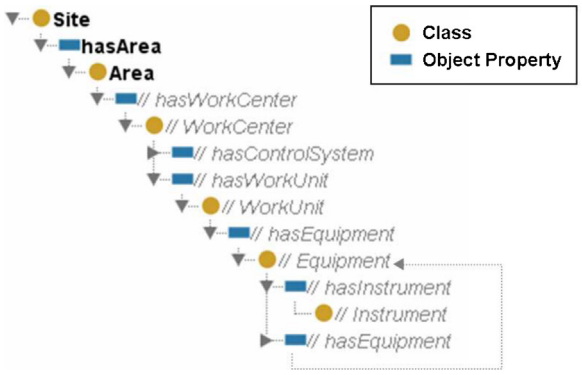


Fig. 2. Fragment of the *Equipment Module* hierarchy specification. Note the link between *WorkCenter* and *ControlSystem* concepts. The *Equipment* class has a recursive property to represent the aggregation structures.

relations. In fact, for the *Equipment* concept, the property becomes recursive, so one equipment can contain others.

Although several design patterns for aggregations has been described in the literature, the descriptive logic implemented by

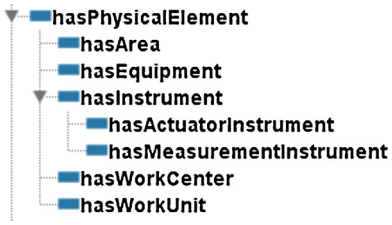


Fig. 3. *hasPhysicalElement*: object properties hierarchy. *hasPhysicalElement* has been defined as a transitive and asymmetric property. These features are inherited by its subproperties.

OWL2 does not provide a differential syntax for this kind of relations. Fortunately, aggregation can be successfully implemented by defining a hierarchy of transitive asymmetric properties as depicted in Fig. 3. Additionally, each property has assigned its appropriate inverse. For instance, *PartOf* is the inverse relation of *hasPhysicalElement*.

With this representation scheme the reasoner is able to deduce many of the new facts bringing a richer information about instances relations.

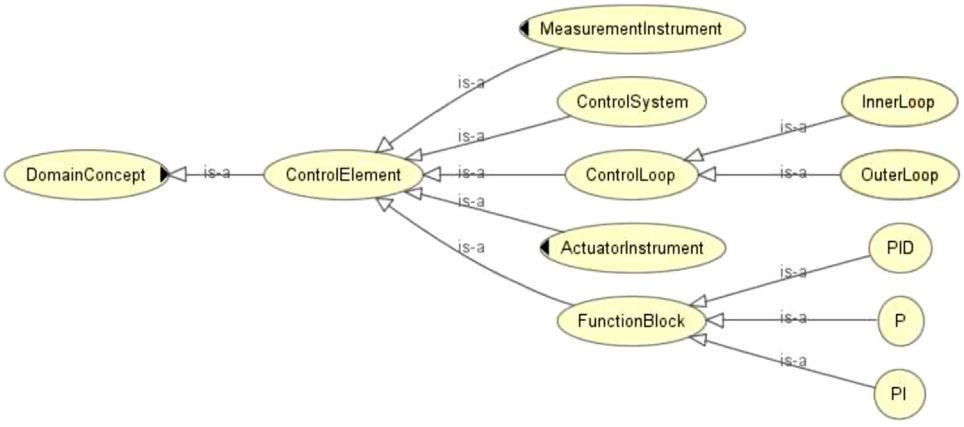


Fig. 4. Jerarquía de clases de elementos de control.

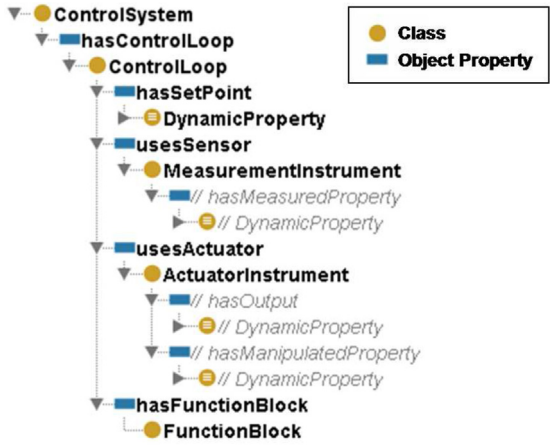


Fig. 5. Outline tree of a classic control loop. Here *hasSetPoint*, *hasMeasuredProperty*, *hasOutput* and *hasManipulatedProperty* are all sub-properties of *hasProperty*.

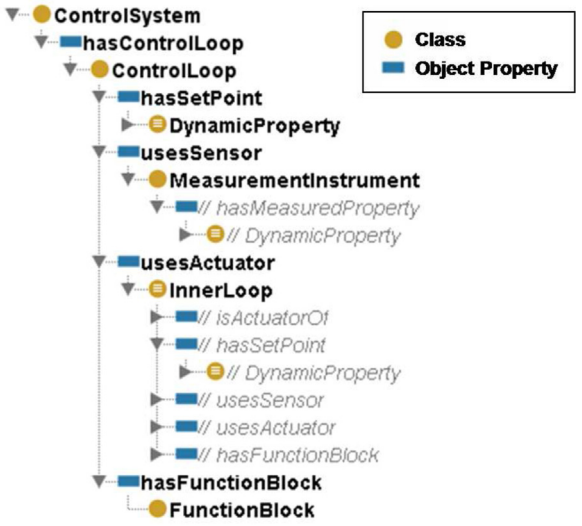


Fig. 6. Outline tree of a cascade control loop. In this control configuration, the range of *usesActuator* property is another control loop.

2.3. Control Module

Control system concepts are organized in a class hierarchy (Fig. 4) tied to the Equipment and Behavioral modules. The top of this hierarchy is the *ControlElement* class, which encloses all the elements that form part of the plant control system. The *ControlSystem* concept itself has been defined as a collection of *ControlLoops*.

Fig. 5 shows the proposed conceptualization of a classic control loop scheme. It has a *FunctionBlock*, a *SetPoint*, some *MeasurementInstruments* and some *ActuatorInstruments*.

Instruments are bound to *DynamicProperties* that represent manipulated or measured process variables. In particular, an actuator instrument is connected with two dynamic properties, the manipulated variable and the command value (e.g. a flow and the control valve opening).

The *FunctionBlock* concept involves possible control functions present in a particular control loop, they are classified in general categories like PI, PD or PID controller. Extensions can be easily included through the definition of new *FunctionBlock* concepts in the hierarchy. MIMO *FunctionBlock* are available for the introduction of multivariable control schemes.

To model a cascade control scheme, the outer loop are connected with the inner loop by a *usesActuator* property as shown in Fig. 6. This particular connection has been used in an equivalent OWL axiom, so that the reasoner can automatically detect cascade configurations, properly identifying *ControlLoops* as *InnerLoops* or *OuterLoop*.

The semantic included in this module makes use of new OWL2 features, that enable the knowledge engineer to specify only some of the relations and the reasoner resolve the missing ones. For instance, the *ControlSystem* is automatically linked with a *WorkCenter* by the locations of its connected *Instruments*. This is achieved through the *hasControlSystem* property – and its inverse *isControlSystemOf*- which has been formally defined as a role inclusion axiom in Description Logic syntax Rudolph (2011). It makes use of a role composition as follow:

$$hasWorkUnit \circ hasEquipment \circ hasInstrument \circ usedBy \circ isControlLoopOf \sqsubseteq hasControlSystem \quad (1)$$

In OWL 2 it is implemented as a property chain axiom. Fig. 7 depicts the represented relation. Here, if the instance *WorkCenter A* has a *WorkUnit B*, *B* has an *Equipment C*, *C* has *Instruments D* which

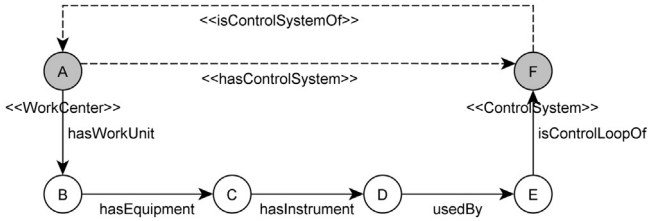


Fig. 7. Relations between *ControlSystem* and *WorkUnit* through a property chain. ≪≫ symbols express inferred classes and properties.

are part of a given *ControlSystem F*, then this property chain states that *A* has a *ControlSystem F* and that *F* is the *ControlSystem* of *A*.

2.4. Supervision Module

The efficient management of measurements is critical to perform on-line process supervision. Plant events (measurements) must be sampled (or triggered), transmitted, stored and processed at a rate in accordance with the process dynamics. If any of these aspects is not properly designed, potential harms could stay hidden for the supervision system. Moreover, information related to measurements represents the major volume of data that the system must manipulate on real time.

In the proposed framework, the system handles events in batches, reasoning only over the latest snapshot and storing the

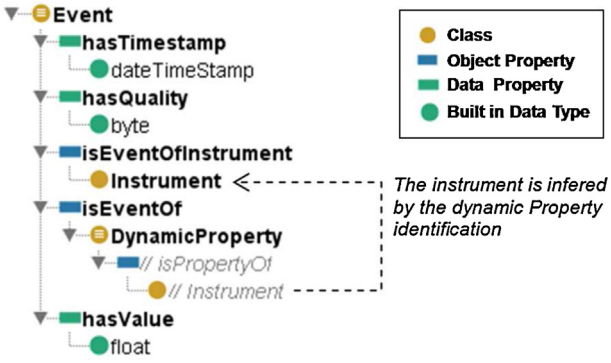


Fig. 8. Event outline tree.

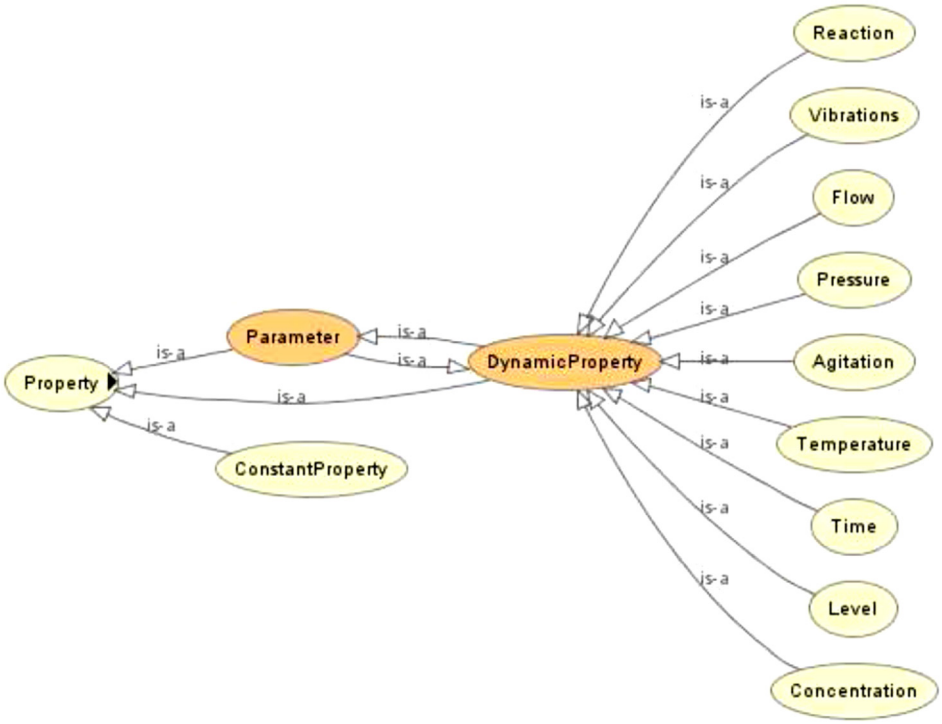


Fig. 9. Property hierarchy.

resulting inferred facts. The event representation has been designed to capture the basic properties for their characterization. As shown in Fig. 8, the *Event* class encloses all the measurements generated by the plant instruments with its values, time-stamps and measurement qualities. A property chain axiom has been included to specify the association between each event and instrument through their associated *DynamicProperty* (i.e. a process variable). A *Property* class is formalized using a two level hierarchy. Categories such as *Temperature*, *Pressure* or *Flow* are defined as disjoint subclasses of *DynamicProperty* (see Fig. 9).

The key concepts of the supervision module has been taken from the HAZOP analysis, which is represented by three interconnected classes: *Deviation*, *Guideword* and *Hazard*. The *Deviation* concept specifies the pattern for which a property value falls into a given guideword. Although in a standard HAZOP these patterns are represented qualitatively, in this implementation pairs of upper-lower boundaries have been specified in order to automatically identify the existence of deviations. Fig. 10 shows the *Deviation* properties considered here.

A deviation is then identified/parameterized by a dynamic variable (or parameter) and a *Guide Word*. Two additional properties are introduced to indicate upper and lower limits associated to a given deviation.

The *Guideword* class is represented following the IEC 61882:2002 standard. General purpose guidewords are established and used to describe possible deviations of process variables (i.e. DynamicProperties). The meaning of these guide words are listed in Table 1. These general expressions are subsequently refined to fit in each variable class (see Table 2 for a complete list of the considered guidewords). In order to capture the semantic of these concepts, every guideword has been formalized with a multiple inheritance scheme.

As an example, Fig. 11 presents the inheritance graph for the *Vacuum* guideword. Thus, the ontology has the knowledge to identify *Vacuum* as a *None* general guide word and as a *Pressure* specific guide word.

The *Hazard* concept links the studied causes, consequences and safeguards related to a particular fault condition. They are obtained from the output of the HAZOP study and should be properly reported by the expert team. Fig. 12 depicts the *Hazard* class and its relations. The *Cause* and *Consequence* concepts describe, respectively, how the deviation may occurs and what may happen in that situation. The *Safeguards* are a list of controls (preventive or reactive) that reduce deviation likelihood or severity. Key relevant rationales, assumptions or data can be captured in a *Comment* field.

The *ActionRequired* concept refers to any required hazard mitigation or control action. A responsible actor to perform these actions can be captured by the *hasActionAssignedTo* property. Finally, the *Risk* class represents a ranking obtained from a combination of the harm probability and its estimated severity. Although *Risk* is not always explicitly identified in HAZOP studies, since the core methodology does not require identification of harms

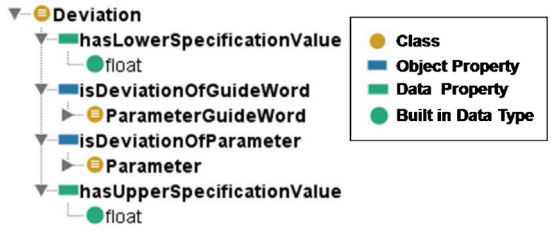


Fig. 10. Outline tree for the *Deviation* concept.

Table 1
Basic guide words and their generic meanings.

Guide word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN	Complete substitution

Table 2
Guidewords multiple inheritances: in the HAZOP analysis, each guideword is associated to general terms and to process variable classes

Parameter/guide word	More	Less	None	Reverse	As well as	Part of	Other than
Flow	High flow	Low flow	No flow	Reverse flow	Deviating concentration	Contamination	Deviating material
Pressure	High pressure	Low pressure	Vacuum		Delta-P		Explosion
Temperature	High temperature	Low temperature					
Level	High level	Low level	No level		Different level		
Time	Too long/too late	Too short/too soon	Sequence step skipped	Backwards	Missing actions	Extra actions	Wrong time
Agitation	Fast mixing	Slow mixing	No mixing				
Reaction	Fast reaction/runaway	Slow reaction	No reaction				Unwanted reaction
Vibrations	Too low	Too high	None				Wrong frequency

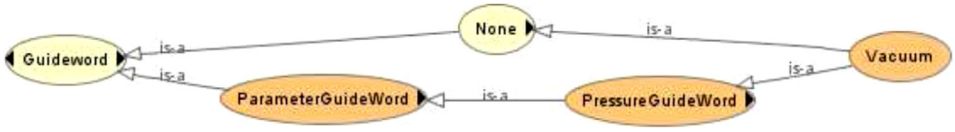


Fig. 11. Multiple inheritance in the guidewords conceptualization. *Vacuum* example.

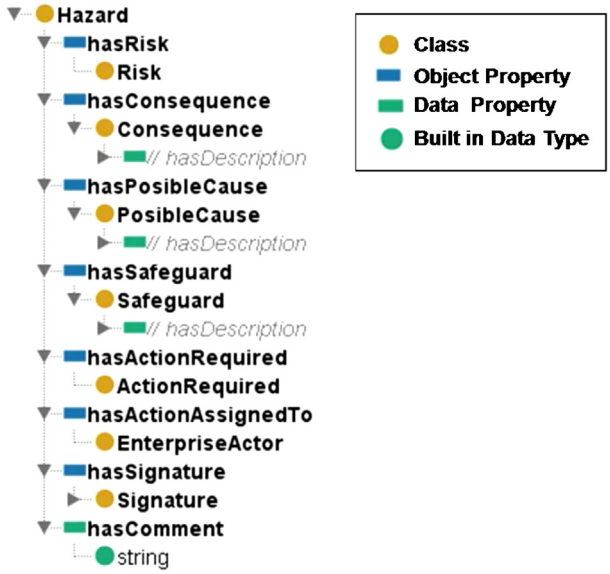


Fig. 12. Hazard concept tree.

probability or severity, assessment teams should rate these factors in order to quantify and prioritize those risks.

In a typical HAZOP, deviations are associated with only one *Hazard*. In the proposed approach, an intermediate concept relates *Hazards* with *Deviations*, without cardinality constrains. This concept is the fault *Signature*, which establishes a deviation pattern (using ranges and guidewords) that leads to the characterization of faulty conditions. Therefore, a *Hazard* will have one or more associated *Signature* instances to support the fault identification process, and each *Signature* will have one or more associated *Deviations*. As a result, it is possible to model a *Hazard* and its links with different dynamic properties deviations building a more powerful HAZOP scheme.

The autonomy of these abstractions brings some remarkable advantages. The conceptual approach facilitates the migration of a developed HAZOP to other similar industrial plants, also supporting the design and analysis of new deviations patterns.

2.4.1. Deviated events and hazard detection

The system detects fault conditions and deduces the possible hazards, bringing the required information to identify and prevent risk situations. This deduction is performed by an inference engine which adds implicit facts to the ontology trough a forward chaining reasoning, from the current events to the hazards analysis.

The object property *hasPossibleDeviation*, which states a possible deviation for a *DynamicProperty* (or its synonym *Paramater*), has been defined as the inverse of *isDeviationOfParameter* shown in Fig. 10. Therefore, if *isDeviationOfParameter* has been properly set when specifying the *Deviation* instance, *hasPossibleDeviation* will be implicitly added by the reasoner. In addition, the following DL role inclusion axiom, is used to propagate the *hasPossibleDeviation* relation from a *DynamicProperty* to its *Events*,

$$isEventOf \circ hasPossibleDeviation \sqsubseteq hasPossibleDeviation \quad (2)$$

The proposed KBS also tests if an *Event* matches with a known *Deviation* through the evaluation of SWRL rules. In particular, the following rules add an implicit fact to bound *Events* to *Deviations* by means of the *isRecognizedAsDeviation* property,

```
hasPossibleDeviation(?e, ?g),
hasLowerSpecificationValue(?g, ?lv),
hasUpperSpecificationValue(?g, ?hv), hasValue(?e, ?v),
greaterThanOrEqual(?v, ?lv), lessThanOrEqual(?v, ?hv)
-> isRecognizedAsDeviation(?e, ?g)
```

In this rule, the variable *e* will be instantiated with a member of the *Event* class, being *v* its value. *g* is a possible *Deviation* defined with limits *lv* and *hv*.

The behavior module includes a *defined class* called *DeviatedEvent* with the following *equivalence axiom* (a necessary and sufficient condition),

$$DeviatedEvent \equiv Event \sqcap (\exists isRecognizedAsDeviation.Deviation) \quad (3)$$

Therefore, the reasoner can automatically group all *Event* members that satisfied the range limits under the *DeviatedEvent* subclass.

A similar classification scheme is applied to identify satisfied deviations, signatures and latent *Hazards*. In order to detect which deviations are implicated in the current events, the *CurrentDeviation* class has been created as a subclass of *Desviation*. *CurrentDeviation* is defined by the axiom,

$$CurrentDeviation \equiv Deviation \sqcap (\exists hasAssociatedEvent.DeviatedEvent) \quad (4)$$

Since *hasAssociatedEvent* was specified as an inverse property of *isRecognizedAsDeviation*, the reasoner is able to perform the right

classification, grouping occurred *Deviations* as members of the *CurrentDeviation* class.

Additionally, the *SatisfiedSignature* class is restricted as follows,

$$SatisedSignature \equiv Signature \sqcap (\forall hasDeviation.CurrentDeviation) \quad (5)$$

Therefore, if a predefined *Signature* is linked with some *Deviations* by the *hasDeviation* property, and they are all members of the *CurrentDeviation* class, the *Signature* will be classified as *SatisfiedSignature*. In other words, a *Signature* is satisfied if all its deviations actually occur.

Note that when evaluating OWL axioms, the system performs forward chaining, classifying instances and inferring new relations. Additionally, if some instances are already classified or properties are defined, it performs a consistency checking.

Finally, as discussed above, a typical *Hazard* member is associated with some fault *Signatures*, so if a *Signature* succeed, the *Hazard* is activated. To represent these state transitions, the *Hazard* class hierarchy has been extended as explained in the next section.

2.4.2. Alarm management

The axiomatic definitions included in the ontology manage alarms associated to each possible *Hazard*. Usually, standard control alarms are set in order to give advice of abnormal variables values when still there is time for process operators to react to the abnormal situation. However, the complexity of modern processes makes difficult the prediction and management of abnormal events since many different things can go wrong in many different ways. Standard control alarms do not give information about root causes of faults, they only inform about some particular deviation. In addition, after the occurrence of an abnormal event, the control system may not be able to work (e.g. they cannot face erroneous measurements). Sometimes, those alarms are so notoriously unreliable or confusing that operators simply ignore them.

In general, supervision monitors deal with three possible alarm states: Active, Inactive, and Acknowledged. Active and Inactive states depend on the existence of a deviated property. Acknowledge is recognized when an operator has taken the responsibility over the abnormal situation.

In order to capture alarms semantic, the current and past alarm states have been analyzed as well as the conditions that cause alarm states transitions. As a result, the *Hazard* concept has been represented in a four level inheritance hierarchy. This scheme permits the access to detailed information about alarm states and activation causes while respecting basic states and rules (see Fig. 13).

In the upper levels there are the basic states *Active* and *Inactive*. As far as we descend in the hierarchy, a more detailed description can be obtained, bringing more information regarding the current situation. This allows controlling the quantity of supplied knowledge according to the business actor, promoting on-demand information access.

The concept *HazardActiveNow* groups every hazard that is active due to the value of current events. These hazards must be informed to the operator for their treatment.

On the other hand, the *HazardActivedPreviously* members are those hazards already informed. To abandon this state, alarms must be acknowledged by operators and the involved variables must return to normal values.

The object property *hasAlarmState* has been added to specify the alarm state of a hazard. It bounds a member of the *Hazard* class with one of the *AlarmState* class. *AlarmState* can take one of three values: *Active*, *Inactive* and *Acknowledged*). *hasCurrentAlarmState* and *hasPreviousAlarmState* are subproperties of *hasAlarmState* that permit to differentiate between the current and previous alarm state.

However, while the current state must be inferred by the reasoner, the previous one had to be set by the previous reasoning cycle or by the intervention of external systems (e.g. acknowledgement using a GUI).

Fig. 14 depicts the alarms states transition diagram associated to a hazard. This diagram was created using UML notation, identifying the alarms states, superstates and transitions. Each transition establishes the conditions that have to be verified in each reasoning cycle. In the figure, the two causes of alarm activation can be observed: the detection of a new fault or the presence of a previous unsolved activation (i.e. the alarm remains active). It is important to note that transitions from the Active to the Acknowledge state cannot be deduced, since it requires the intervention of an external agent.

Two approaches to formalize these Alarm state transitions have been evaluated. In one hand there is the rule-based approach that in this framework could be implemented by SWRL. In the other hand, the transitions logic can be formalized by a proper class hierarchy and DL axioms to define the semantic of each one. Since axioms keep the rules integrated to class definitions, the later approach enables a better processing by DL reasoners. Additionally, it exploits the expressive capability of OWL2 leading to considerable reductions of reasoning times. Therefore, in order to implement the conditions that trigger alarm states transitions in accordance with Fig. 14, each class hold *Equivalence* and *SubclassOf* axioms. The former defines the conditions that must be fulfilled by the members of the *Hazard* class and are used by the reasoner to perform a proper instance classification on the hierarchy shown in Fig. 13. The *SubclassOf* axioms set the proper alarm states corresponding to each *Hazard* subclass.

$$\begin{aligned}
&HazardActivatedPreviously \equiv hasPreviousAlarmState.\{active\}\\
&HazardActivatedPreviously \sqsubseteq ActiveHazard \sqcap hasCurrentAlarmState.\{active\}\\
&HazardActiveNow \equiv \exists hasSignature.SatisfiedSignature\\
&AcknowledgeHazard \equiv HazardActiveNow \sqcap (hasPreviousAlarmState.\{acknowledged\})\\
&AcknowledgeHazard \sqsubseteq hasCurrentAlarmState.\{acknowledged\}\\
&UnacknowledgeHazard \equiv HazardActiveNow \sqcap (hasPreviousAlarmState.\{inactive\})\\
&UnacknowledgeHazard \sqsubseteq hasCurrentAlarmState.\{active\}\\
&InactiveHazard \equiv Hazard \sqcap \neg ActiveHazard\\
&InactiveHazard \sqsubseteq hasCurrentAlarmState.\{inactive\}
\end{aligned} \quad (6)$$

3. Knowledge exploitation on the Tennessee Eastman process

The proposed approach has been applied to the Tennessee Eastman (TE) process.(Downs & Vogel, 1993). This benchmark provides a rich configuration of equipment and control loops that enables the validation of the main functionality of the proposed KBS.

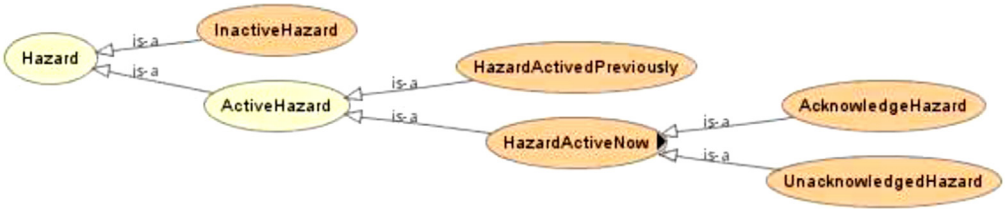


Fig. 13. Hazard inheritance hierarchy.

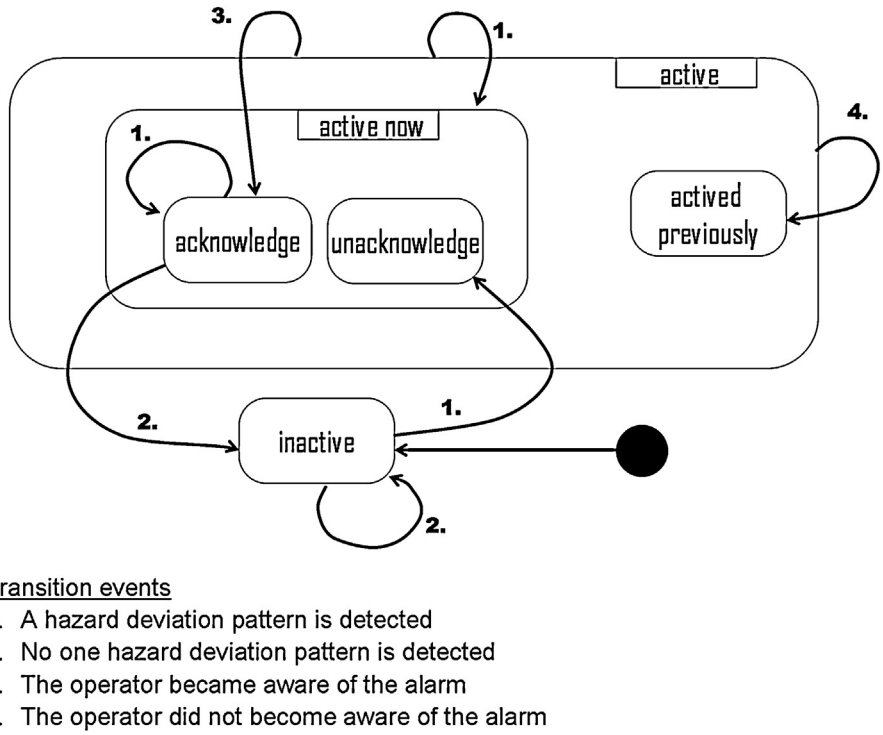


Fig. 14. Alarm state transition diagram.

The TE involves the production of two gas products; *G* and *H*, from four liquid reactants, *A*, *C*, *D* and *E*. Additionally, there occur two side reactions and an inert *B*. All reactions are irreversible and exothermic. The process has five main units: an exothermic 2-phase reactor, a product condenser, a flash separator, a reboiled stripper and a recycle compressor.

To proceed with the behavioral tests, a partial HAZOP analysis was developed analyzing the process disturbances presented by Downs and Vogel (1993). Disturbances have been simulated in the interval $[2h, 50h]$ and measured with a sample time of $T_s = 10s$. The complete set of data includes 41 measured process variables and 12 actuators signals. For each disturbance, the corresponding variable deviations were identified. Simulation tests were performed using the code provided by Braun and Rivera (2011), and the decentralized control system proposed by McAvoy and Ye (1994) (see Fig. 15).

The first step in this study is to represent the TE process concepts consistently with the proposed ontological specification. Therefore, the classes and relations has been instantiated generating explicit facts on the Control, Equipment and Supervision modules. This representation required the incorporation of 135 individuals, 117 object property assertions, and 20 data property assertions. Part of this information is extracted in Tables 3–6. It is worthy to note that on a real process, this information can be obtained combining several data sources, such as the plant information system, control systems, and P&ID diagrams.

Table 3
TE actuator instruments list (Extract).

Instrument	Description	Equipment	Manipulated variable
CV_{12}	Reactor coolant valve	Reactor	$F_{RC} = XMV(10)$
CV_1	A feed valve	Mixer	$F_A = XMV(3)$
CV_2	D feed valve	Mixer	$F_D = XMV(1)$
CV_3	E feed valve	Mixer	$F_E = XMV(2)$
CV_{13}	Condenser coolant valve	Reactor	$F_{CC} = XMV(11)$
AG_1	Reactor agitator	Reactor	$V_A = XMV(12)$

The second step is reasoning. The chosen reasoner was Pellet 2, because it has some required capabilities no present in other reasoners, such as complex data-type reasoning and SWRL rules support. Following a W3C practice, the instances have been separated from the conceptual representation, and were distributed in two OWL files. An incremental reasoning strategy has been designed to obtain good performance on real time. It may be defined as a two-stage process,

Stage 1: The reasoner checks the ontology consistency using the axioms present in the KB itself – It is one of the advantages of this paradigm because it facilitates data sharing – and makes inferences over the low frequency changing instances that experience relatively few changes (e.g. equipment, control configuration, etc.). This reasoning is performed under user demand.

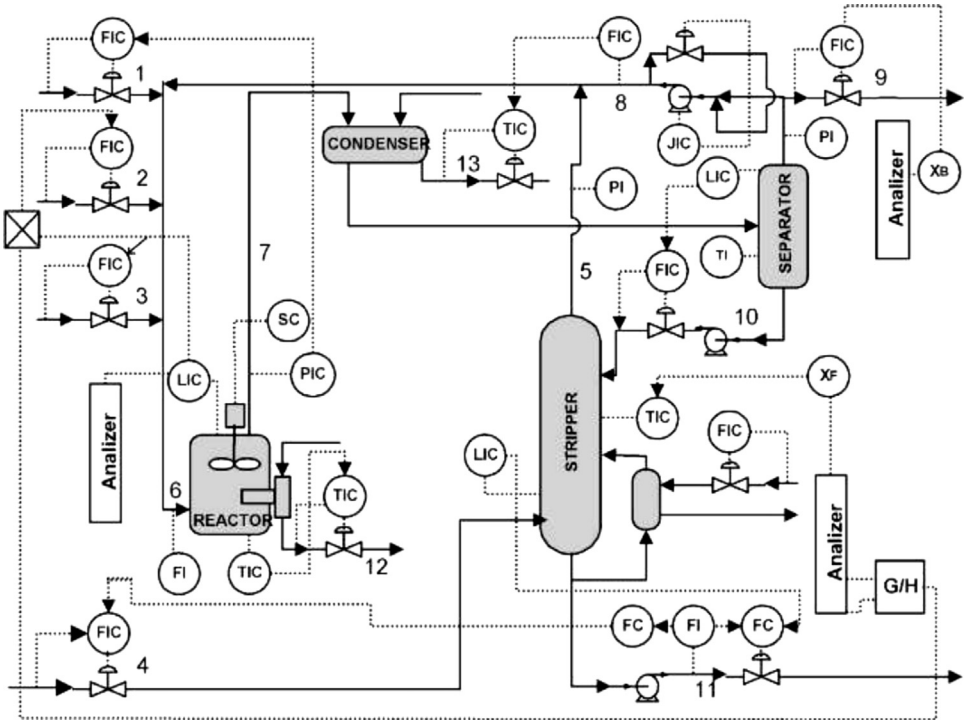


Fig. 15. Tennessee Eastman process flowsheet.

Table 4
TE measurement instruments list (Extract).

Instrument	Description	Equipment	Measured variable
TI_{12}	Reactor coolant temperature indicator	Reactor	$T_{RC} = XMEAS(21)$
TI_{13}	Condenser coolant temperature indicator	Condenser	$T_{CC} = XMEAS(22)$
PI_7	Reactor pressure indicator	Reactor	$P_R = XMEAS(7)$
TI	Reactor temperature indicator	Reactor	$T_R = XMEAS(9)$
FI_1	A feed flowmeter	Mixer	$F_1 = XMEAS(1)$
FI_2	D feed flowmeter	Mixer	$F_2 = XMEAS(2)$
FI_3	E feed flowmeter	Mixer	$F_3 = XMEAS(3)$
FI_4	A and C feed flowmeter	Mixer	$F_3 = XMEAS(4)$
FI_6	Reactor intake flowmeter	Reactor	$F_6 = XMEAS(6)$
FI_8	Recycle flowmeter	Mixer	$F_8 = XMEAS(5)$
LI	Reactor level indicator	Reactor	$L_R = XMEAS(8)$
CA_1	Reactor intake concentrations indicator	Reactor	$X_A - X_F = XMEAS(23 - 28)$

Table 5
TE control cascade inner loops list (Extract).

Control loop	Measurement instrument	Actuator instrument	Set point	Function block
FC_1	FI_1	CV_1	F_{1SP}	PI
FC_3	FI_3	CV_3	F_{3SP}	PI
TC_1	TI_{12}	CV_{12}	T_{12SP}	PI
xFC_2	FI_2	CV_2	F_{2SP}	PI
xTC_2	TI_{13}	CV_{13}	T_{13SP}	PI

Table 6
TE control loops list (Extract).

Control loop	Measurement instrument	Inner loop	Set point	Function block
ReactorPC	PI_7	FC_1	P_{RSP}	PI
ReactorLC	LI	FC_3	L_{RSP}	PI
ReactorTC	TI	TC_1	T_{RSP}	PI
xRecycleFC	FI_8	TC_2	F_{8SP}	PI

Stage 2: This reasoning is performed automatically on real time. The system uploads to the KB the newest process data, at a sample time on the order of a few seconds, according to the process dynamic. Using these data and the results of previous reasoning cycles, the reasoner adds inferred facts about the dynamic instances (measurements, alarms, etc.). This stage is summarized in Fig. 16. First, a DL reasoner checks if the current *Events* match with its *Possible Deviations*. When coincidence, *Events* are classified as *DeviatedEvent* and the correspondent *Deviation* as *CurrentDeviation*. Then, the reasoner compares the resulting deviation pattern

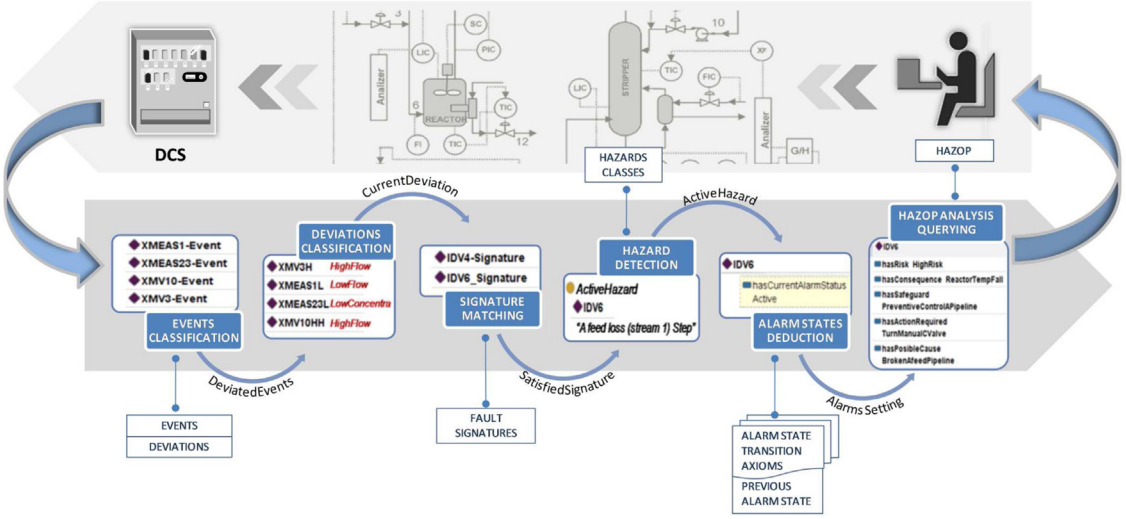


Fig. 16. Illustration of forward chaining reasoning in the Tennesse Eastman process.

with the stored fault *Signatures*. The matching signatures are classified as *SatisfiedSignature*. Then, *Hazards* associated with these signatures are immediately identified, but their classification in the ontology depend on the alarm states. Hence, using the *previous alarm states* and the axioms depicted in (Eq. (6)), the reasoner infers the *current alarm states* and places *Hazards* in the proper subclass. Finally, the system brings relevant information using stored *HAZOP* knowledge.

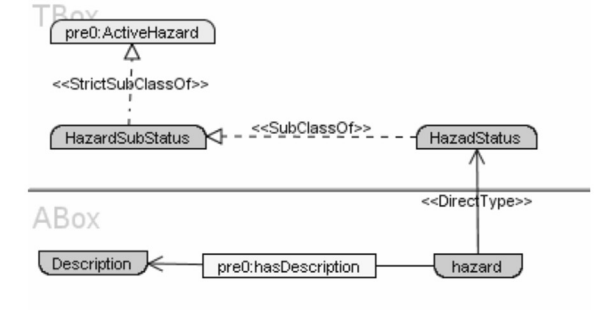
To evaluate this strategy, the reasoning test has been performed 50 times in an Intel I5-750 PC with 3 Gb RAM memory. The average reasoning time was 4.8 s. and 1.8 s. for the first and second stages respectively.

After reasoning, the stored and inferred knowledge can be retrieved in three different ways: navigation, search (i.e. navigation + goal), and query (i.e. explicit question).

In this environment, querying the ontology requires a language that recognizes RDF as the fundamental syntax. SPARQL was chosen because it is a W3C recommendation with a simple and powerful syntax. SPARQL queries can be expressed as RDF graph patterns that the processor matches with the stored axioms. The service to run these queries is provided by and endpoint that accepts and processes SPARQL. In this case, OWL2 query tab v1.0 was mounted over Protégé to perform queries in the ontology instantiated with the TE concepts.

Queries have been designed to answer competency questions concerning the supervision tasks, thus supporting a process analysis with regards to particular abnormal conditions.

The RDF graph pattern of Fig. 17a is used to exemplify the detection of active *Hazards*. Labeled arcs represent relations that are matched with the knowledge base properties assertions. Dark grey nodes represent variables that will be instantiated with explicit or inferred individuals. Type restrictions are indicated in the upper panel while the property pattern between class members are indicated in the lower one. This can be translated into the SPARQL query of Fig. 17b. Fig. 17c depicts the retrieved results. *Hazards*



(a)

```
PREFIX onto:<http://www.semanticweb.org/PSOntology.owl>
SELECT ?Description ?Type ?hazard
WHERE
  { ?hazard onto:hasDescription ?Description;
            <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?Type.
    ?Type   <http://pellet.owldl.com/ns/sdle#strictSubClassOf> onto:ActiveHazard.
  }
```

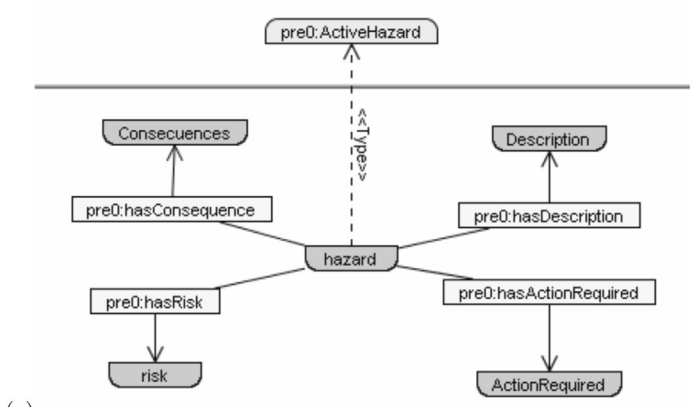
(b)

Results

?hazard	?Description	?HazardState	?HazardSubstate
pre0:IDV7	"C header pressure loss-reduced availability (stream 4) "^^xsd:stri...	pre0:HazardActivedPreviously	pre0:HazardActivedPreviously
pre0:IDV6	"A feed loss (stream 1) Step"^^xsd:string	pre0:HazardActiveNow	pre0:UnacknowledgedHazard
pre0:IDV4	" Reactor cooling water inlet temp. Step"^^xsd:string	pre0:HazardActiveNow	pre0:AcknowledgeHazard

(c)

Fig. 17. Hazard detections. (a) RDF query pattern. (b) SPARQL Query. (c) Results table.



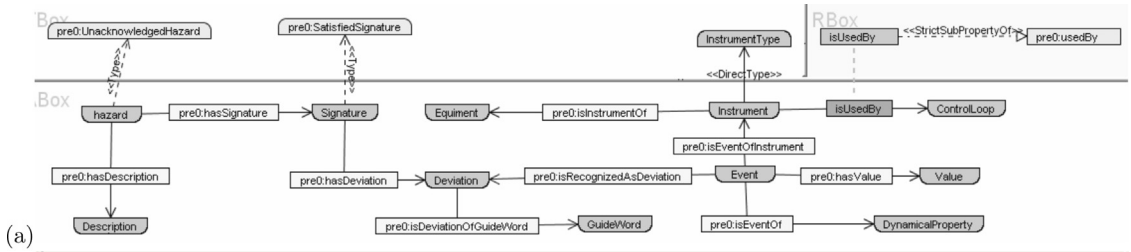
(a)

(b)

Results

?hazard	?Description	?risk	?Consecuences	?ActionRequired
pre0:IDV6	"A feed loss (stream 1) Step"^^xsd:string	pre0:HighRisk	pre0:ReactionRateFall	pre0:TurnManualCValve
			pre0:ProductOutOfSpecification	pre0:CheckAValvePipe
			pre0:ProductivityLost	
			pre0:ReactorTempFall	
			pre0:FlowReductionADE	
			pre0:C_ConcentrationRice	
			pre0:ALeak	
pre0:IDV4	" Reactor cooling water inlet temp. Step"^^xsd:string	pre0:HighRisk	pre0:WholePlantDisturbance	pre0:CheckCoolantSupplyProcess
			pre0:HighConsumptionOfCoolant	
			pre0:ProductivityLost	

Fig. 18. Active hazards. (a) RDF query pattern. (b) Results table.



(a)

(b)

Results

?hazard	?Description	?Value	?DynamicalProperty	?GuideWord	?Instrument	?InstrumentType	?Equiment	?isUsedBy	?ControlLoop
pre0:IDV6	"A feed loss (stream 1) Step"...	"6"^^xsd:float	pre0:XMV3	pre0:HighFlow	pre0:CV-1	pre0:ActuatorInstrument	pre0:Mixer	pre0:usedAsActuatorBy	pre0:FC1
pre0:IDV6	"A feed loss (stream 1) Step"...	"-4"^^xsd:float	pre0:XMEAS1	pre0:LowFlow	pre0:FI-1	pre0:MeasurementInstrument	pre0:Mixer	pre0:usedAsSensorBy	pre0:FC1
pre0:IDV6	"A feed loss (stream 1) Step"...	"-5"^^xsd:float	pre0:XMEAS23	pre0:LowConce...	pre0:CA-1	pre0:MeasurementInstrument	pre0:Reactor	pre0:usedAsSensorBy	pre0:none

Fig. 19. Equipment involved in an active hazard. (a) RDF query pattern. (b) Results table.

*IDV*4 and *IDV*6 have been activated because the current events have a pattern that can be associated with the deviations established in the identification procedure. Moreover, since their previous states were *Acknowledge* for *IDV*4 and *Inactive* for *IDV*6, they were also subclassified as *Acknowledge* and *Unacknowledge* respectively. *IDV*7 presents a previous faulty condition that was not acknowledged by the operator, so its alarm state remains *Active*.

To support operator decisions under hazardous conditions, the KBS must provide additional information coming from the HAZOP analysis. Fig. 18 shows the query that returns information about the causes, consequences and risk level associated with any hazard considered in active state by the system.

In front of a risk situation, the KBS supplies information about the involved equipment. The query depicted in Fig. 19 enables to trace the active *Hazard* starting from its activation facts, including deviated events, the instruments that made the measurements and the equipments to which the instruments belong. Moreover, the system shows the involved control loops and the role performed by each instrument in them (i.e. sensor or actuator). Fig. 19b shows the results of this query, retrieved while IDV6 is present.

All this information is a valuable asset to help operators to make timely and efficient decisions.

4. Concluding remarks

In this work, an ontology-based framework for process supervision in chemical plants has been presented.

A conceptualization of equipment, control systems and hazards, have been developed. It includes the semantic of each term in order to obtain a heavyweight ontology that have been formalized using Description Logic. A knowledge-driven approach has been adopted in order to demonstrate how DL reasoning could be used to support process supervision without the help of external agents.

In the proposed approach a DL reasoner adds implicit facts to the ontology, reasoning from the current process measurements to the characterization of hazardous states. Additionally, the system is able to perform consistency checking and formally explain the obtained results.

The system provides the needed flexibility to capture and manage incomplete information, enabling fast adaptations and implementations. Moreover, the KB can be extended, becoming more precise and useful, only by the addition of new concepts and relations without modifying existing software applications.

The ontology has been implemented with OWL2, which is based on the SROIQ Description Logic, allowing richer reasoning deductions and a superior consistency checking.

The proposed ontology acts as meta-data, so that the semantic assertions are mounted over existing data sources making it richer and meaningful. To support plant operators decisions, key knowledge can be retrieved using a SPARQL engine. A valuable characteristic of the presented system, is that it is capable of explain its outputs, since they were obtained through logical deductions done by the reasoner.

Future work will consider historical data, in particular to perform trend analysis in order to further improve the dynamical capabilities of the fault diagnosis system. Also, the alignment of this ontology with the included in Intelligent HAZOP tools will be investigated.

Acknowledgement

Financial support from CONICET, ANPCyT and FCEIA-UNR is fully acknowledged by the authors.

References

Batres, R., Aoyama, A., & Naka, Y. (2002). A life-cycle approach for model reuse and exchange. *Computers and Chemical Engineering*, *26*(4–5), 487–498.

Batres, R., & Naka, Y. (2000). Process plant ontologies based on a multi-dimensional framework. In M. F. Malone, & J. A. B. C. Trainham (Eds.), *Fifth international conference on foundations of computer-aided process design* (pp. 433–437).

Bhagwat, A., Srinivasan, R., & Krishnaswamy, P. (2003). Fault detection during process transitions: A model-based approach. *Chemical Engineering Science*, *58*(2), 309–325.

Brandl, D., Broersen, S., Chappell, D., Charpentier, L., & Craig, L. (2007). ISA Draft TR 88/95.00.01, batch control and enterprise-control system integration, using ISA-88 and ISA-95 together. Tech. rep., ISA.

Braun, N., & Rivera, D. (2011). Tennessee Eastman problem for matlab. Available from http://csel.asu.edu/?q=node/33 Accessed December 2011.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2006). Ontological engineering: Principles, methods, tools and languages. In C. Calero, F. Ruiz, & M. Piattini (Eds.), *Ontologies for software engineering and software technology* (pp. 1–48). Berlin, Heidelberg: Springer.

Downs, J., & Vogel, E. (1993). A plant-wide industrial process control problem. *Computers and Chemical Engineering*, *17*(3), 245–255.

Dunjó, J., Fthenakis, V., Vílchez, J. A., & Arnaldos, J. (2010). Hazard and operability (HAZOP) analysis. A literature review. *Journal of Hazardous Materials*, *173*(1–3), 19–32.

Gernaey, K., & Gani, R. (2010). A model-based systems approach to pharmaceutical product-process design and analysis. *Chemical Engineering Science*, *65*(21), 5757–5769.

Ghosh, K., Ng, Y., & Srinivasan, R. (2011). Evaluation of decision fusion strategies for effective collaboration among heterogeneous fault diagnostic methods. *Computers and Chemical Engineering*, *35*, 342–355.

Gillett, J. (1997). *Hazard study and risk assessment in the pharmaceutical industry: John Edward Gillett*. Buffalo Grove, IL: Interpharm Press.

Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Hebeler, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2009). *Semantic web programming*. Indianapolis, IN: Wiley Publishing Inc.

ISA. (1995). ANSI/ISA-S88.01 – 1995, Batch control – Part I: Models and terminology.

ISA. (2000). ANSI/ISA-95.00.01 – 2000, Enterprise/control system integration – Part 1: Models and terminology.

Krötzsch, M., Simancik, F., & Horrocks, I. (2012). A description logic primer. CoRR abs/1201.4089.

Leveson, N. G. (1995). *Safeware: System safety and computer*. New York: Addison-Wesley Publishing Company.

MacGregor, J., & Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. *Computers and Chemical Engineering*, *47*, 111–120.

Marquardt, W., Morbach, J., Wiesner, A., & Yang, A. (2010). *OntoCAPE: A re-usable ontology for chemical process engineering* (Rwth ed.). Springer.

Maurya, M. R., Paritosh, P. K., Rengaswamy, R., & Venkatasubramanian, V. (2010). A framework for on-line trend extraction and fault diagnosis. *Engineering Applications of Artificial Intelligence*, *23*(6), 950–960.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2007). Fault diagnosis using dynamic trend analysis: A review and recent developments. *Engineering Applications of Artificial Intelligence*, *20*(2), 133–146.

McAvoy, T., & Ye, N. (1994). Base control problem for the Tennessee Eastman problem. *Computers and Chemical Engineering*, *18*(5), 383–413.

Moore, R., & Kramer, M. (1986). *Expert systems in on-line process control*. Expert Systems in Process Control.

Morbach, J., Wiesner, A., & Marquardt, W. (2009). Onto cape-a (re-) usable ontology for computer-aided process engineering. *Computers and Chemical Engineering*, *33*, 1546–1556.

Musulin, E., Yelamos, I., & Puigjaner, L. (2006). Integration of principal component analysis and fuzzy logic systems for comprehensive fault detection and diagnosis. *Industrial and Engineering Chemistry Research*, *45*(5), 1739–1750.

Muthuswamy, K., & Srinivasan, R. (2003). Phase-based supervisory control for fermentation process development. *Journal of Process Control*, *13*(5), 367–382.

Natarajan, S., Ghosh, K., & Srinivasan, R. (2012). An ontology for distributed process supervision of large-scale chemical plants. *Computers and Chemical Engineering*, *46*, 124–140.

Ould-Bouamama, B., El Harabi, R., Abdelkrim, M., & Ben Gayed, M. (2012). Bond graphs for the diagnosis of chemical processes. *Computers and Chemical Engineering*, *36*, 301–324.

Parmar, J., & Lees, F. (1987a]). The propagation of faults in process plants: Hazard identification for a water separator system (part ii). *Reliability Engineering*, *17*(4), 303–314.

Parmar, J., & Lees, F. (1987b]). The propagation of faults in process plants: Hazard identification (part i). *Reliability Engineering*, *17*(4), 277–302.

Patton, R., & Chen, J. (1997). Observer-based fault detection and isolation: Robustness and applications. *Control Engineering Practice*, *5*(5), 671–682.

Qian, Y., Li, X., Jiang, Y., & Wen, Y. (2003). An expert system for real-time fault diagnosis of complex chemical processes. *Expert Systems with Applications*, *24*(4), 425–432.

Ram Maurya, M., Rengaswamy, R., & Venkatasubramanian, V. (2004). Application of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets. *Engineering Applications of Artificial Intelligence*, *17*(5), 501–518.

Rudolph, S. (2011). Foundations of description logics. In *Reasoning web. Semantic technologies for the web of data*. Berlin: Springer.

Ruiz, D., Nougués, J. M., & Puigjaner, L. (2001). Fault diagnosis support system for complex chemical plants. *Computers and Chemical Engineering*, *25*, 151–160.

Srinivasan, R., Wang, C., Ho, W., & Lim, K. (2005). Context-based recognition of process states using neural networks. *Chemical Engineering Science*, *60*(4), 935–949.

Stanford Center for Biomedical Informatics Research. (2012, February). Protégé. http://protege.stanford.edu

Sundarraman, A., & Srinivasan, R. (2003). Monitoring transitions in chemical plants using enhanced trend analysis. *Computers and Chemical Engineering*, *27*(10), 1455–1472.

Vaidhyanathan, R., & Venkatasubramanian, V. (1995). Digraph-based models for automated HAZOP analysis. *Reliability Engineering and System Safety*, *50*(1), 33–49.

Vaidhyanathan, R., & Venkatasubramanian, V. (1996). A semi-quantitative reasoning methodology for filtering and ranking HAZOP results in HAZOPExpert. *Reliability Engineering and System Safety*, *53*, 185–203.

Venkatasubramanian, V., Rengaswamy, R., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis, Part II: Qualitative model and search strategies. *Computers and Chemical Engineering*, *27*, 313–326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., & Yin, K. (2003). A review of process fault detection and diagnosis, Part III: Process history based methods. *Computers and Chemical Engineering*, *27*, 327–346.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis, Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, *27*, 293–311.

Venkatasubramanian, V., & Rich, S. (1988). An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis. *Computers and Chemical Engineering*, *12*(9–10), 903–921.

Venkatasubramanian, V., & Vaidhyanathan, R. (1994). A knowledge-based framework for automating HAZOP analysis. *AIChE journal*, *40*(3), 496–505.

Venkatasubramanian, V., Zhao, J., & Viswanathan, S. (2000). Intelligent systems for HAZOP analysis of complex process plants. *Computers and Chemical Engineering*, *24*(9–10), 2291–2302.

Villa, F., Athanasiadis, I., & Rizzoli, A. (2009). Modelling with knowledge: A review of emerging semantic approaches to environmental modelling. *Environmental Modelling and Software*, *24*(5), 577–587.

Villez, K., Rosén, C., Anctil, F., Duchesne, C., & Vanrolleghem, P. A. (2013). Qualitative Representation of Trends (QRT): Extended method for identification of consecutive inflection points. *Computers and Chemical Engineering*, *48*, 187–199.

Villez, K., Srinivasan, B., Rengaswamy, R., Narasimhan, S., & Venkatasubramanian, V. (2011). Kalman-based strategies for fault detection and identification (FDI): Extensions and critical evaluation for a buffer tank system. *Computers and Chemical Engineering*, *35*, 806–816.

Wang, Z., Zhao, J., & Shang, H. (2012). A hybrid fault diagnosis strategy for chemical process startups. *Journal of Process Control*, *22*(7), 1287–1297.

Wiesner, A., Morbach, J., & Marquardt, W. (2011). Information integration in chemical process engineering based on semantic technologies. *Computers and Chemical Engineering*, *35*(4), 692–708.

Yu, J. (2012). Local and global principal component analysis for process monitoring. *Journal of Process Control*, *22*(7), 1358–1373.

Zhang, M., Ge, Z., Song, Z., & Fu, R. (2011). Global–local structure analysis model and its application for fault detection and identification. *Industrial and Engineering Chemistry Research*, *50*(11), 6837–6848.

Zhao, C., Bhushan, M., & Venkatasubramanian, V. (2005a]). PHASuite: An automated HAZOP analysis tool for chemical processes, Part I: Knowledge engineering framework. *Process Safety and Environmental Protection*, *83*(6), 509–532.

Zhao, C., Bhushan, M., & Venkatasubramanian, V. (2005b]). PHASuite: An automated HAZOP analysis tool for chemical processes, Part II: Implementation and case study. *Process Safety and Environmental Protection*, *83*(6), 533–548.

Zhao, J., Cui, L., Zhao, L., Qiu, T., & Chen, B. (2009). Learning HAZOP expert system by case-based reasoning and ontology. *Computers and Chemical Engineering*, *33*(1), 371–378.

Zhao, Y., Jiang, C., & Yang, A. (2012). Towards computer-aided multiscale modelling: An overarching methodology and support of conceptual modelling. *Computers and Chemical Engineering*, *36*, 10–21.