HAL open science

Anti-logicist framework for design-knowledge representation

Antonio Giovannini, Alexis Aubry, Hervé Panetto, Hind Bril El-Haouzi, Ludovic Pierrel, Michele Dassisti

To cite this version:

Antonio Giovannini, Alexis Aubry, Hervé Panetto, Hind Bril El-Haouzi, Ludovic Pierrel, et al.. Anti-logicist framework for design-knowledge representation. Annual Reviews in Control, 2015, 39 (1), pp.144-157. 10.1016/j.arcontrol.2015.03.013 . hal-01122807

HAL Id: hal-01122807

https://hal.science/hal-01122807v1

Submitted on 4 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Anti-logicist framework for design-knowledge representation

A. Giovannini [a,b,c]. A. Aubry [a,b]. H. Panetto [a,b]
H. El Haouzi [a,b]. L.Pierrel [c]. M. Dassisti [d]

[a] CNRS, CRAN UMR 7039, France
[b] Université de Lorraine, CRAN UMR 7039, Boulevard des Aiguillettes
B.P. 70239 F-54506 Vandœuvre-lès-Nancy, France
(e-mail: {antonio.giovannini; alexis.aubry; herve.panetto; hind.el-haouzi}@univ-lorraine.fr)
[c] TRANE SAS, rue des Amériques, 88190 Golbey, France (ludovic_pierrel@trane.com)
[d] Politecnico di Bari, Viale Japigia 182, 70126, Bari, Italy (michele.dassisti@poliba.it)

Abstract: The knowledge reuse and mapping are among the most important concerns related to the design knowledge representation. In this paper, authors focus on the importance of one specific property of a design knowledge representation: the unambiguity. Authors show 1) how the ambiguity of the representation can increase the risk of a failure in the reuse and mapping processes, 2) how most of works in the literature use formal logic constructs and finally 3) how the use of these can increase the risk of ambiguity. On the basis of these remarks, an overview on the works on the anti-logicist architecture is provided: the systems based on this architecture show an intelligent behaviour without using logic constructs. An analysis and a transposition of the anti-logicist principles are then performed to build a framework allowing to represent design knowledge without logic constructs. To do so 1) main concepts are formalised in a conceptual model; 2) an algorithm has been designed to map pieces of knowledge based only on the representation syntax; 3) two instantiations of the framework are showed using a CAD instantiation. Finally, the limits of the current deployment of the framework and the research perspectives are discussed.

Keywords: Knowledge representation, Design, Anti-logicism, Knowledge reuse, Conceptual modeling, Computer aided design

1. INTRODUCTION: THE AMBIGUITY OF THE REPRESENTATION AS MAIN CONCERN

Mapping and reuse of knowledge from the customer's domain onto the product domain is one of the main issues of the product line design (Giovannini et al., 2014). The mapping and the reuse of the design knowledge models are the main concerns to make automatic and/or to speed up the design stages. This paper focuses on the issues concerning a general knowledge mapping and reuse independently on the specific couple of domains to be mapped.

Both the reuse and the mapping involve an effective communication of the represented knowledge from a modeller to potential users; to this aim, in this work, authors postulate the following statement:

> *S1 - A knowledge modeller (M) should build some design-knowledge model (K) to make the potential users (Ds) — e.g. product designer, process designer — able to use it — i.e. to meet the customer of the design stage — exactly in the same way the modeller was intended to do.*

In this paper, authors propose an approach of the design-knowledge representation to satisfy the statement S1, namely, the *unambiguity of the knowledge representation from the modeller's and the user's points of view*. An ambiguous representation is subject to more than one possible interpretation. In presence of ambiguity, it is very difficult to be sure that all the potential users of the model will interpret it in the same way as the modeller intended. This uncertainty implies that, when a massive knowledge communication is necessary, the inferences based on the modelled knowledge are highly unreliable. For instance, let us consider a design change in a complex system: in order the design process to be efficient, all the knowledge about the system to be changed should be understood unambiguously to be mapped and reused (using the knowledge required to modify it). In presence of ambiguity, modeller-user interactions are necessary to make the representation usable, since the modeller is required to explain his interpretation to the users of the knowledge. Ambiguity is a de-facto lack in the completeness of the model that needs to be supplied by the user interpretation.

Many research efforts have been done in the knowledge representations (KR) domain so far. Most of works use formal logics, i.e. formal logic constructs, to build knowledge models. Formal logic uses the natural language in a structured fashion (from Fredge and Peano in (Van Heijenoort, 1977)): in this paper, authors show how this is the point that can increase the risk of ambiguity of the represented knowledge. As an effect of this remark, an investigation of approaches that do not use logics has been performed, i.e. *anti-logicist* constructs. Contributions found mainly focus on robotics and multi-agent systems (Agre &

Chapman, 1987; Brooks, 1991; Kaelbling, 1987). The provided examples of *anti-logicist* systems (e.g. multi-agents, robots) present an intelligent behaviour without a representation based on logics (and so without the use of natural language). Therefore, on the basis of the *anti-logicist* works, an original analysis and transposition to the design KR domain of the main *anti-logicist* principles is here proposed. This allowed us to propose a framework for the design KR based on the *anti-logicism*.

The main idea is that the measuring systems become part of the design-knowledge model. Since the measuring systems are considered as the only way designers can represent the reality (i.e. perceive and react on it), the KR model is explicitly connected with how the designer is interfaced with the reality. Measurements should replace the role of the abstract concepts in logicist KR. Abstract concepts need a user interpretation in order to be connected with the reality and so to make the knowledge usable. Therefore, to avoid user interpretations (i.e. ambiguity), everything should be represented as measurements and mathematical relations between them. As consequence, every two KR models based on two different sets of measuring systems are considered as describing the behaviour of two distinct systems.

In summary, the proposal is a framework for the design-KR that does not use logic constructs and so does not use natural language. As a result, the main feature of the knowledge model built based on the proposed framework should be the *unambiguity*, i.e. since the connection with the reality is explicit in the model, the user do not need a personal interpretation to use the represented knowledge.

The structure of the paper is as follows. In the next section an overview of the *logicist* and *anti-logicist* approaches is presented, comparing these based on the possible ambiguity of the representation. In section 3, an analysis of the anti-logicist principle is performed. The proposed framework is also presented using a conceptual model, an explanatory case instantiation of the framework and an algorithm to show the unambiguity of the representation. In section 4, a method to instantiate the framework onto a CAD environment is presented. A validation test is provided in section 5 for the design process of a heating, ventilation and air-conditioning (HVAC) system component.

2. ON THE DESIGN KNOWLEDGE REPRESENTATION: LOGICS AND AMBIGUITY

The aim of this section is to provide a critical overview of the logic-based representation available in the scientific literature so far. Based on the impacts of the use of logics on the ambiguity of the representation, the *logicist* approaches are compared with the *anti-logicist* ones, i.e. approaches of KR that do not use *logics*.

2.1. Logic-based Knowledge Representation (KR)

One of the most used methods for design-knowledge representation is the Description Logic (DL). DLs were developed to bridge the gap between the previous techniques for the KR and the formal logics (e.g. first-order logic - FOL) (Baader, Horrocks, & Sattler, 2008). In general, DLs are representations based on formal logics and are used to describe concepts belonging to specific application domains. First applications of DLs were decidable, i.e. the reasoning process on the representation never ends in infinite loops. Indeed, most of DLs are decidable parts of FOL (Baader et al., 2008; Grosof, Horrocks, Volz, & Decker, 2003; Schild, 1991).

As for DLs, other approaches can be used for KR by representing relations between abstract concepts at different level of expressiveness and reasoning complexity. Here is below a summary of those approaches for KR:

- *Boolean satisfiability:* these provide a generic combinatorial reasoning and a search platform. These are based on propositional logic: a propositional or Boolean formula is a logic expression over variables that can take true or false values (Gomes, Kautz, Sabharwal, & Selman, 2008).
- *Constraint programming* (CP): it is another paradigm that provides combinatorial reasoning. CP is based on techniques derived from artificial intelligence, operational research, graph theory and others. CP formalises analytical and logical relations between variables (Rossi, Van Beek, & Walsh, 2008). In most of the successful implementation, variables are usually defined in finite domains.
- *Conceptual graphs:* these are representations based on the semantic networks of artificial intelligence and the existential graphs of Charles Sanders Peirce. The graphs semantics can be mapped with FOL. Moreover the graphical definition of the knowledge can be an advantage for humans (i.e. friendly representation) as well as for machines (i.e. the regular structure of the graph should simplify algorithms for reasoning, indexing, searching and pattern matching). (ISO/IEC, 2007, p. 24707; Sowa, 2008)
- *Qualitative modelling* is mainly an attempt to operate with as minimum knowledge as possible. High levels of abstraction are required; as a consequence the risk of ambiguity of the representation is high too. This makes qualitative models a human-friendly complement to traditional numerical techniques. (Forbus, 2008)
- *Model-based problem solving* is a paradigm that aims to extend the applicability of expert rules (e.g. if X then Y) on more than one domain. A knowledge model about certain generic systems should support the representation of the conditions for the applicability of the rules that should describe particular tasks. The representation of the knowledge has to be based or linked to *logics*. (Struss, 2008)
- *Bayesian networks:* these are a tool for modelling and deciding about uncertain conditions. Bayesian networks associate a qualitative with a quantitative component: a graph qualitatively represents the relations between possible events and conditional probabilities to quantify their influences (Darwiche, 2008). The graph nodes represent conceptualisation of particular condition or events.

2.2. Critics about logics in literature

All the techniques reviewed in the above paragraph are directly or indirectly based on the use of *logics*. But different

remarks have been formulated about the use of logics for KR. These remarks focused on the following four points (a more detailed discussion, from a *logicist* viewpoint, is in (Lifschitz, Morgenstern, & Plaisted, 2008)):

- *Reasoning about exceptions*. The modelling and the reasoning process about exceptions can be a really tough task for logic-based KR. As an example, refer to the following proposition:

if (x is bird) and (is not a penguin) then (x can fly).

Here there are two issues about the formalisation of the exceptions: 1) there are too many exceptions; 2) if it is known that *y* is a bird that is not enough to infer that *y* can fly. (Lifschitz et al., 2008)

In order to cope with these criticalities, *non-monotonic* logic-based reasoning was developed. The main idea about this can be synthesized as follows: "in the absence of any information to the contrary, assume…" (Brewka, Niemelä, & Truszczynski, 2007). Reasoning mechanism that include non-monotonic reasoning and find a solution in a reasonable amount of time are widely discussed in (Gelfond, 2008).

- *Reasoning based on logics is too expensive*. According to the critics of *logicism*, reasoning about several thousands or millions of axioms is simply not possible. Several researches in logics stressed on this point. Even if a lot of efforts in the last years have been done to improve the reasoning performances, some good results have been obtained under restrictive assumptions (Lifschitz et al., 2008).

- *Other approaches do it better*. An *anti-logicist* critic is based on the existence of other methods that can represent knowledge better than *logics*. According to *logicists*, logic-based method remain the best way to represent knowledge for complex application and other approaches can be better for the solution of simpler problem, e.g. learning with a restricted vocabulary (Lifschitz et al., 2008).

- *Representing all the knowledge is infeasible*. Even if there have been successful implementations of the *logicist* approaches, some doubts about the following criticisms still remain: it is difficult to become aware of all the implicit knowledge and then make it explicit; there is some apparently obvious knowledge that is really difficult to express in any language (Davis, 1998); there are no efficient theory about the reasoning about the absence of knowledge; a conceptual model for a domain is a prerequisite and represents something extremely difficult to build; the mapping problem is a quite hard issue because concepts do not always match easily (Lifschitz et al., 2008).

2.3. About anti-logicist corpus of knowledge

The first scientific work proposing an alternative to the *logics* for KR were published in mid-80s: the works of Agre and Chapman on a computer program called Pengi (Agre & Chapman, 1987), the Brooks' subsumption architecture (Brooks, 1991) and the Kaelbling's intelligent reactive systems (Kaelbling, 1987) are examples of this approach to KR. The main ideas behind these solution approaches are the following:

- intelligent behaviour can be obtained without representation and reasoning about symbols, i.e. natural language (Brooks, 1991);
- intelligence emerges from the interaction between the system and the environment (Brooks, 1991);
- perception directly triggers reactions, that means no reasoning intervenes between each reaction (Agre & Chapman, 1987);
- systems react without having an explicit model of the world (Agre & Chapman, 1987);
- things in the world are not represented objectively; that is to say independently of the system's purpose, they are represented in relation with the system nature and its projects (Agre & Chapman, 1987).

In summary, *anti-logicists* do not formalise explicit plans to be executed by a system. They give the system a way to interact with the environment (e.g. sonar, motors), a set of skills (i.e. how to avoid an object, how to explore an environment) and a set of purposes (e.g. do not hit objects, explore). On the basis of the perceptions, one or more skills are applicable. A prioritization system decides which one have to be applied at a certain time.

2.4. Anti-logicism as an alternative to natural language ambiguity

Now, how *logicism* or *anti-logicism* can deal with the statement S1? Let us consider a customer and a designer that agree on the requirements of the design stage, i.e. **X**. The designer (*U*) has not the knowledge to meet **X** therefore he should retrieve a model *K* that describes the necessary knowledge. *K* should describe how *U* should react on the reality (i.e. designer's degrees of freedom, **Y**) in order to meet **X**, i.e. how to deploy the proper production stage. In order to have an *unambiguous K*, *U* should be able to get the same **Y**, as the modeller of *K* would have got. Any other potential user of *K* should be able to validate both the conditions **X** and the results **Y** as well. This validation is related to how the users perceive and impact the reality, i.e. **X** (e.g. sensors, measuring devices) and **Y** (e.g. actuators, motors, product and process parameters) (Fig.1). Therefore, in order to *K* not to need any user interpretation, the modeller should link every part of *K* to the user capabilities of perceptions and reactions, i.e. **X** and **Y**. If the modeller is able to model *K* by using **X** and **Y** that all users share, then *K* is not ambiguous. Notice that, if users have different **X**s and **Y**s then several *Ks* should be represented (i.e. one for each set of capabilities). Indeed, since the model has to include the representation of **X** and **Y**, different **X**s and **Y**s naturally relate to distinct *Ks*.

About logicist representation. One of the original purposes of logic was the transformation of the natural language in something more precise and clear to be computed formally like equations in algebra (from Fredge and Peano in (Van Heijenoort, 1977)). Therefore each logic-based representation requires the use of the natural language, i.e. at least those words that define concepts. Concepts are not directly related to the perception/reaction capabilities, i.e. they are not of the same nature. As a consequence, each logic-based representation requires a user interpretation to build the

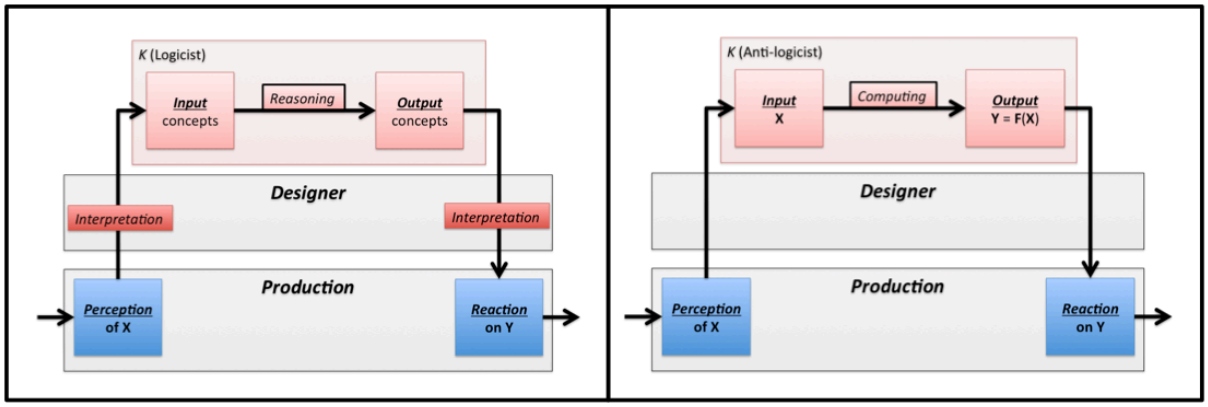

Fig.1 - Representation of how the *logicist* and *anti-logicist* knowledge models are used.

concept/perception and concept/reaction links (Fig.1). And thus, there is no-guarantee that two users provide the same interpretation. Therefore the knowledge to be used by a generic user can be ambiguous because of its own interpretation.

Moreover, a critical point still remains even if the considered case is when the modeller links all the concepts with some perception/reaction capabilities (i.e. concepts are expressed as set of values of some measured properties). Indeed, as explained above, logic has been developed with the aim of reasoning with the natural language and not with numerical values.

About anti-logicist representation. The shown *anti-logicist* architectures for robots and multi-agent systems have the following common features: the designed architectures 1) show intelligent behaviours without the use of logic and so natural language (Brooks, 1991) 2) provide a direct links of the knowledge model with the perceptions of the designed systems (as clearly shown in Fig.1). Therefore if a *K* is represented according to *anti-logicist* principles, the interpretations of the users are not required to use *K*. Therefore *K* could result to be *unambiguous*.

3. ANTI-LOGICIST FRAMEWORK FOR KNOWLEDGE REPRESENTATION

The aim of this section is to transpose the *anti-logicist* principle into the design KR domain to propose the framework. The framework is expected to provide 1) the underlying structure of an *anti-logicist* design-KR and 2) the impacts on the design knowledge mapping and reuse. The concepts and an explanatory case are presented in the subsections 1, 2 and 3. A design simulation and a mapping algorithm provide an overview of the knowledge mapping and reuse in the proposed framework (in subsections 4, 5, 6).

3.1. Analysis of the anti-logicist principles

In *anti-logicist* systems, the systems react on the basis of some perceptions of the world collected by sensors. Knowledge models are directly connected to the information collected by the sensors. Knowledge models connect this information to the reaction that the system can express in their environment, i.e. actuators (e.g. movement by means of motors). Therefore the models are about the *perception-reaction* relations, i.e. the description about how to alter the *system-environment* interaction in order to achieve a system goal. Both input and output of the knowledge model are directly connected to sensors and actuators those provide the perceptions and reactions. Therefore the model is complete because it is expressed in terms of the same properties measured by the sensors and impacted by the actuators. Therefore no-interpretation is required to use the represented knowledge. In other words, there is no conceptual definition of the knowledge that should be instantiated to infer an appropriate system reaction to lead the system toward its goals.

Doing so, the *anti-logicist* research outcomes show that the use of the represented knowledge is possible without reasoning about concepts described by using natural language (Fig.1). If the KR is achievable without using these concepts, then also a representation of the design knowledge without interpretations would be theoretically possible.

In order to build an anti-logicist KR framework, the following transposition of concepts is required (see Tab. 3.1). In the proposed framework for the unambiguous design KR, the knowledge models should be rooted in the reality, i.e. perceptions by sensors and reactions by actuators. For design knowledge modellers, perceptions and reactions should be enacted through *measurements* (say, temperature, humidity, etc.).

For *anti-logicist* systems, the perception-reaction relations allow then to orient the system towards its goals. For design knowledge modellers, the relations between *measurements* allow to describe to users how the represented knowledge can meet certain customer requirements. For *anti-logicist*, this approach allows intelligence to emerge from the system-environment interactions. In this framework, this approach should allow the result of the design stage to emerge from the knowledge models that links customer requirements (i.e. **X**) and designer's degrees of freedom (i.e. **Y**). Connecting the knowledge to the *measurements* (as the connection to the sensors for the *anti-logicist* systems) should guarantee the completeness of the model, i.e. the measurement are directly linked to the input/output of the knowledge model and so no-further interpretations of the knowledge are required to use it. Therefore even if the potential users are experts in different domains (issue detailed in (Whitman & Panetto, 2006)), a correct communication (see statement S1) of the represented knowledge should be performed.

Tab. 3.1 - Transposition of concepts.

Anti-logicist systems	Anti-logicist KR
Sensors and actuators (i.e. **X** and **Y**)	Measurements (i.e. **X** and **Y**)
Perception-Reaction models	*Requirements - degrees of freedom* models
Intelligence without *logics*	Use of the knowledge without ambiguity

3.2. Main concepts of the proposed framework

In the framework here proposed, the *measurement* is the objective concept that allows to avoid the user interpretation and so the consequent ambiguity in the KR (as the *perception* concept does for the *anti-logicists*).
The conceptual model showing the basic concepts of this framework is shown in Fig. 2. The semantics of the main concepts in the proposed framework is the following.

- A *measurement* is the characterisation of the act of perceiving, by mean of a measurement device, a certain measure in a certain environment at a given time. For instance, the temperature of a given volume of air at a given time is a possible *measurement*.
- Each *measurement* is identified univocally by a vector *(S, T, S)*: *space*, *time* and *shape* characterisation. For instance, the temperature can be the *shape* of the *measurement* (i.e. *what* to measure), the volume of air can be the *space* of the *measurement* (i.e. *where* to measure) and the *time* represent *when* the temperature is measured. Each one of these three elements are defined as *properties*.
- Each *property* is involved in one or more *transformations*: a *transformation* is a mathematical relation between a set of *properties*. For instance, the relation between the temperature and the time in which it is measured is the *transformation* $T=f(t)$, where t is the time of the *measurement*, T is the temperature and f is the mathematical relation that links the two *properties*.
- A set of mathematical relations between the properties of a set of *measurements* is defined *experience*. An *experience* is a sort of report of an observation (i.e. experiment): here the setup (*properties* that are constant during all the observations) and the variables observed during the experiment are all formalised as mathematical relations between *measurements*. As an example, let us consider an experiment to test the validity of the ideal gas law $PV=nRT$ (an instance of *transformation*). The instance of the *experience* has to capture the mathematical relations between all the (S, T, S)s of the *measurements* (e.g. the pressure, the temperature, the volume of the gas considered) that describe the environmental conditions where the law holds. In other words, the *experience* instance describes a law and when this law is applicable. All this is formalised by *measurements* and *transformations*.
- Each *property* has to be detailed by a *range of values*, a *UOM* (unit of measure) and a *tolerance* of the *measurement*. For instance, the *shape* of the *measurement* can be measured in Celsius degrees (*UOM*) between 25°C and 50°C (*range of value* for the validity of the experimental law) with a *tolerance* of ±0.1.

The main idea behind this KR framework is that every atom of knowledge is an *experience* and every component of the *experience* should be perceivable by a measuring system. The connection of every atom of knowledge with the measurements should guarantee the unambiguity (similarly to the perception for *anti-logicist*). The instances of *transformation* in an *experience* represent the behavioural laws of the described system and the characterisation of the applicability of the laws. This applicability is here described by the *UOMs*, the *ranges of values* in which the system behaviour has been observed and the *tolerances* (error of the sensor) of every *property* of each *measurement*.
Note that even the observed objects are characterised by *measurement* instances. For instance a copper cable should be described as a cylinder (*space*) in which it is possible to measure some *shapes* that should define the interaction of the copper with the rest of the observed system, e.g. the conductivity of the copper if the behaviour of an electric power system is the object of the observation.
This approach allows to formalise all objects (that are typically formalised as abstract concept in *logicist* approaches) as *measurements* that are directly connected with the system behaviour and thus with knowledge expressed by the instances of *experience*. This provides a mean to avoid any user interpretation to connect the perception of the reality (here described by the *measurements*) with the knowledge about a system behaviour (here described by the *transformations* in each *experience*). As above explained, if there is no need of instantiate abstract concepts, the designer interpretation is not required to meet some customer requirements **X**.
The framework deals with all the knowledge that can be representable as measurements, e.g. even the customer requirements are *measurements*. Even for concepts hard to imagine as measurable — because strictly related to the human perceptions — there exist scientific corpus of knowledge available to formalise mathematically physical-physiological relations (Chauvet, 1993; Lieber, Dupont, Bouffaron, & Morel, 2013).

3.3. Explanatory case

For the sake of simplicity, here, authors will refer to a real industrial example (less complex than the one in the next section) to show the design KR instantiation by using the framework proposed: let be the designer's ***experience*** the representation of the heat exchange phenomenon between a certain volume of air and the flow of water in a bare-tube coil. Here a given volume of water flows into the coil at a given interval of time to cool the airstream. To instantiate the proposed model, the *space* characterisation of the *measurements* is required.

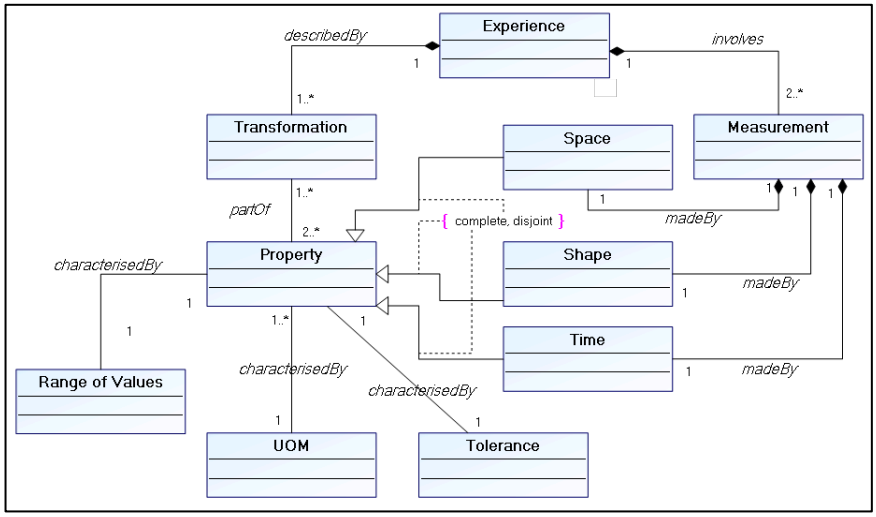

Fig. 2 - Conceptual model of the framework

This can be easily performed by using a CAD software: parameters are added to the CAD files and linked to the volumes or surfaces that should provide the representation of *where* the *measurement* is performed. In the case of dynamic behaviour, the *shapes* (i.e. variables such as temperature, pressure) and the *spaces* (i.e. volumes) can be expressed as function of the *time* variable. The software here adopted to illustrate the framework instantiation is CATIA V5 by Dassault Systems[1]. The details about the usage of the software can be found in the fourth section, before the application of the proposed framework on the case study. The representation in CATIA of the bare tube coil system is shown in Fig.3: the volumes occupied by the copper (i.e. tubes), the water (i.e. that flows in the tubes) and the air (i.e. in front of and behind the coil) are represented. The represented behaviour is a sensible heat transfer as in (ASHRAE, 2012).

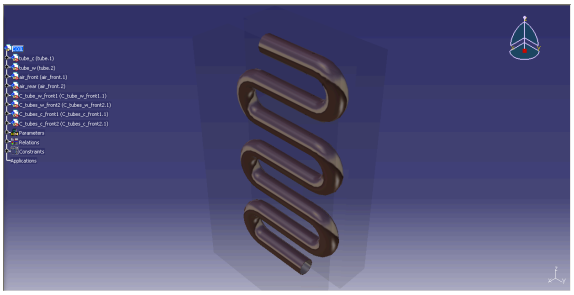

Fig.3 - Representation of the bare tube coil *experience* on CATIA V5.

The instances of the ***measurement*** concept are:

- the temperatures of the air before and after the heat exchange, T_A1 and T_A2;
- the temperatures of the water before and after the passage in the coil, T_R1, and T_R2;
- the speed of the air, V_A;
- the density and the specific heat capacity of the air and the water, RO_A, RO_W, C_P, C_R;
- the film coefficient of the heat transfer between the air and the external coil surface, F_A;
- the film coefficient of the heat transfer between the water and the internal coil surface, F_R.

The relative ***shapes*** are the temperatures, the speed, the density and the heat transfer coefficients. All measurements are related to specific volumes or surfaces (the ***space*** characterisation). For instance, in Fig.4, the volume related to the C_P, RO_A and T_A1 measurements is the one behind the coil. The link of the measurement *spaces* with the volumes represented in the CAD model is performed adding a specific parameter and associating it to the measurement of the relative volume or surface. For instance, the parameter vol_T_A2 and the *formula.155* (shown on the left of the Fig.5) associate the *space* characterization of the T_A2 *measurement* with the volume represented in the CAD.

The observed system is stationary and thus all measurements are ***time***-invariant. The associations of the *measurements* to the volumes provide a characterisation of the properties related to the volume that are involved in the system behaviour. For instance, the association of the C_P and RO_A with the air volumes characterise the property of the air that are involved in the behaviour of the system, i.e. the sensible-heat transfer.

Therefore the abstract concept of *air* is formalised by mean of the (S, T, S)s of the C_P and the RO_A. In this way this framework provides a representation of the object without using the natural language. The abstract concepts that should define an object are replaced by the *shapes* of the *measurements* in the volume occupied by the object in a certain time. The *shapes* that define the object are the properties that characterise the interactions between the objects in the represented system. In other words, objects are represented by mean of the *properties* that are involved in the system behaviour described by the *transformations*.

Because of any knowledge is defined by measurements, no interpretations are needed to connect a certain measured reality with a modelled system behaviour. Therefore no

[1] http://www.3ds.com/products-services/catia/products/v5/

ambiguity is possible when the formalised knowledge will be retrieved and interpreted for designing a new system. Some more details about how experiences connect each other's will be provided in the next section.

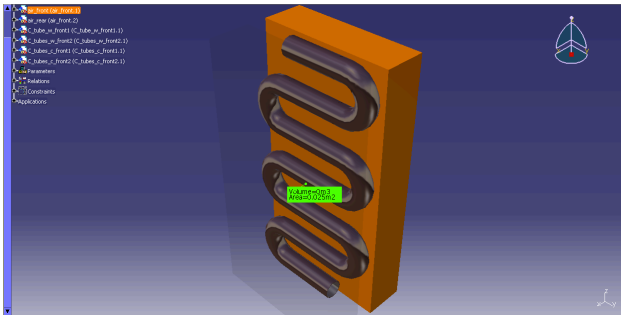

Fig.4 – Representation of the bare tube coil *experience*: the volume behind the coil is related to the C_P, RO_A and T_A1 measurements.

The ***transformations*** between the (S, T, S)s of the measurements are shown on the left of Fig.5. The mathematical relations (i.e. *transformations*) include a group of properties not related to the above cited measurement:

- Q_T is the heat exchange rate;
- W_A is the air mass flow;
- W_R is the water mass flow;
- DELTA_T is the mean temperature difference between the airstream and the water;
- U_0 is the overall coefficient of heat transfer for sensible cooling (without dehumidification).

Indeed, these properties can be derived by mathematical relations between the above measurements; e.g. the DELTA_T is calculable from the water and air temperature values. The other properties not cited in the transformations are:

- A_A that is the front surface of the coil;
- A_0 is the external surface of the coil;
- A_I is the internal surface of the coil.

These three properties are directly related and calculated on the basis of the characterisation of the coil geometry in the CAD model.

Actually also F_A and F_R can be calculated on the basis of the coil geometry, but the detailed instantiation is out of the scope of this section. A more comprehensive instantiation can be found in the fourth paragraph.

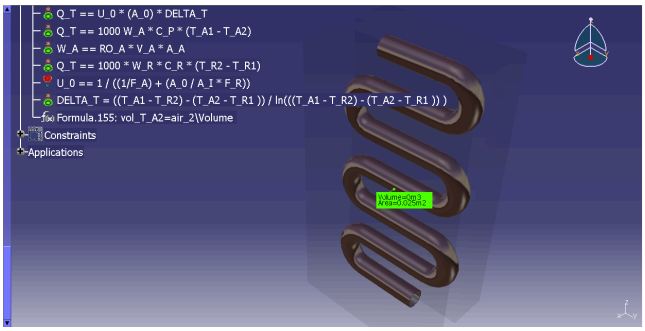

Fig.5 - Representation of the bare tube coil *experience*: the transformations in CATIA V5.

3.4. How the experience instances can be retrieved and connected in a product/process design stage

Since all the KR is based on the definition of the *measurements*, the *experiences* have to be retrieved and mapped on the basis of the *measured* (S, T, S). Since this definition guarantees the non-ambiguity of the representation, the connection should not be subject to more than one interpretation of the knowledge user, either human (say customers, engineers) or machine.

Customer-to-Product

An experience can be retrieved by defining a set of requirements (**X** in Fig.1) as *measurement* (S, T, S)s. Each property has to be defined in *UOM*, *range of values* and *tolerance*. The requirements constrain the *experience* compatibility.

For instance the bare-tube coil design experience can be retrieved if a customer define a certain temperature value and tolerance to be measured in a given volume. The customer should be asked for other *properties* values: this should characterise the constraints that have to be respected by the designer, e.g. the volume occupied by the coil. For instance, the volume can be related to other measurements (e.g. RO_A, C_P) that characterise the presence of the air in the volume. All the remaining properties become degrees of freedom for the engineer.

In order to have the compatibility of the *experience* with the requirements, the customer defined temperature must have a value included in the *range of values* described by the *experience*. Moreover, the *tolerance* in the experience needs to be equal or more precise than the one of the temperature requirement, i.e. if the customer asks for 25±0.01 the experience to satisfy the requirement should be related to an observation with at least the same precision.

Product-to-Process

If the coil is not physically available (i.e. in stock), an experience to manufacture the coil is required. Therefore the *measurements* related to the engineer degrees of freedom (**Y** in Fig.1) become the new requirements for the coil manufacturing *experience*. As for the air, the coil is a volume characterised by certain measurements, e.g. the thermal conductivity that impacts the values of the F_R and F_A and so the thermal exchange. The manufacturing *experience* is represented as an alteration of these *measurements* in time, e.g. the deformation of the copper bended by the appropriate machine.

In summary, the knowledge retrieval is performed on the basis of a (S, T, S) correspondence, i.e. a temperature or a thermal conductivity at a certain *time* in a certain *space*.

3.5. How to test that a solution exists for the defined customer requirement

Once the set of *measurements* and the set of *transformations* that describe the system are available, a model to design a product and process to meet the customer requirements is available. The customer and the process are represented by *properties* so they are characterised by a *range of values*, a *UOM* and a *tolerance*. Customer-related *properties* (characterisation of **X** in Fig.1) are the *property* values constrained by the customer requirements. Process-related

properties (characterisation of **Y** in Fig.1) represent the degrees of freedom that engineers have to manufacture the products that should meet the customer requirements.
The presence of a value for the *tolerance* allows to build the ranges of values that a *property* can take to respect 1) the precision of the formalised knowledge (i.e. the precision of the *experience*-related measuring systems) and 2) the degree of tolerance of the customer about the defined properties values. For instance, let us consider that only the property *x* is representing the customer. The property *x* has values of 15 with a tolerance of ±1. Then, the range of values that can take x to satisfy the customer is described by

$$14 < x < 16 \quad .$$

This means that each value included in this ranges cannot be appreciated by the measuring systems of the customer. The retrieved product/process knowledge is expected to be at least as precise as the customer's measuring system.
Once the ranges for the customer and the process properties are defined, a test to verify the feasibility of a process solution for a customer needs to be performed. This test is a mathematical problem in which the constraints are represented by the *transformations* involved in the connection of *measurements* in the domain of the customer (i.e. requirements, **X**) with *measurements* concerning manufacturing processes (i.e. degrees of freedom of the production, **Y**). A customer-process feasibility test is required for each *customer-process* couple. In order to structure a customer-process feasibility test, a Mixed Integer Non-Linear Problem (MINLP). Notice that, MINLP is required to provide flexibility to the formalisation of design knowledge, i.e. the expressiveness of the framework should be not restricted to linear relations between continuous variables. The general model is the following:

$$\begin{aligned}
&find\, \bar{x}, \bar{y}, \bar{k} \\
&s.t. \\
&L_1 < \bar{x} < U_1 \\
&L_2 < \bar{y} < U_2 \\
&L_3 < \bar{k} < U_3 \\
&f_j(\bar{x}, \bar{y}, \bar{k}) = 0, j \in [1, P] \\
&g_w(\bar{x}, \bar{y}, \bar{k}) \le 0, w \in [1, Q]
\end{aligned}$$

where:

- the *P* functions *f* and the *Q* functions *g* are the linear and/or non-linear constraints representing the *transformations* (equations in Fig.5 for the bare tube coil example),
- the vector *x* has dimensions representing the *properties* related to the customer (e.g. the air temperature to be kept constant in the above example),
- the vector *y* has dimensions representing the *properties* related to the process (e.g. the bending force to apply on the copper tube to deform it),
- the vector *k* has dimensions representing the *properties* involved in the *transformations* but not belonging neither to the customer definition nor to the process definition (e.g. the coil geometry can be completely defined constraining the impact on customer requirements and the parameters of the coil manufacturing process),
- L_i and U_i are the vectors with the lower and upper bounds for each dimension of x ,y ,k (i.e. for x and y the bounds are calculated using the tolerances as above; for k bounds represent the range of values that the properties can assume and they remain invariant during the feasibility tests of all the possible x and y values).

Variables can be integer or real.
Each model can be solved by an appropriate MINLP solver. The solver has to verify that a solution respecting all the constraints and the properties ranges exists. If a solution exists then the considered manufacturing process (i.e. set of values for the properties that define the process) is able to manufacture a product that meets the requirements of the considered customer (i.e. set of values for the properties that define the customer).

3.6. A mapping based on the syntax

Since the represented knowledge is expected to be unambiguous, in this section an algorithm is built to compare two different experiences only on the basis of the syntax of the model. This should demonstrate that even in mapping the relations between two pieces of knowledge, no-interpretation of the representation is required.
The algorithm should allow to understand, ***by syntax comparisons (i.e. no-required user interpretation)***, if either 1) one experience includes the other or 2) the experiences are *inconsistent* or 3) the experiences are not related. Therefore this algorithm should allow to map the relations between different pieces of knowledge and/or different knowledge bases.
The algorithm designed in this paper is used to show how the comparison between two different experiences can be performed without human interpretation and so it can be automated. The aim is not to present an algorithm for the automatic mapping. The actual purpose is to prove the non-ambiguity of the representation by developing a mapping process completely based on the syntax of the model, i.e. the values of the (S, T, S)s.
In Fig.6, a flowchart representing the algorithm to compare a couple of *experiences* is shown. The most important points of the algorithm are the following:

- *experiences* can have all the (S,T,S) compatible for each *measurement* (at least on a certain range of values, e.g. experience 1 has X from 10 to 20 and experience 2 to has X from 15 to 22 then experiences are compatible from 15 to 20); in this case if the *transformations* are equivalent (same values for all corresponding properties of the two experiences) then the *experiences* represent the same piece of knowledge (conclusion 2); otherwise there is an inconsistency (conclusion): a possible explanation is that one influential

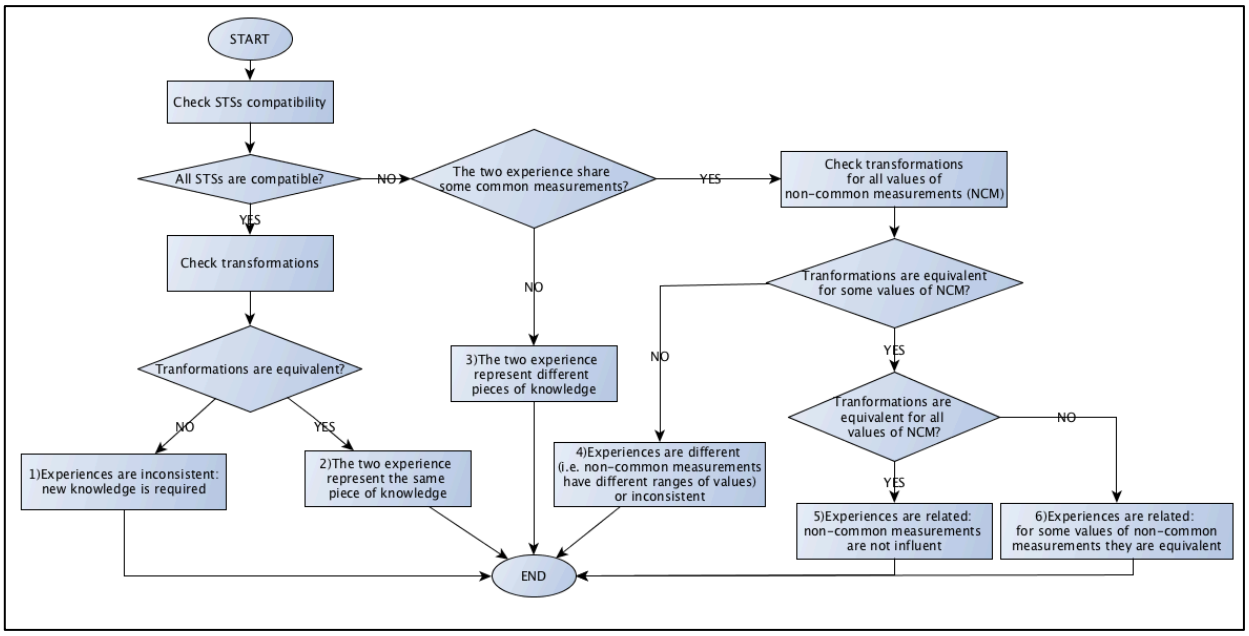

Fig.6 - Flowchart representing the algorithm presented in the current section.

factor has not been observed during the *experience*-related experiments;

- *experiences* have to be (S,T,S) compatible, i.e. all *UOM*, *tolerances* and *ranges of values* for *shapes*, *times* and *spaces* for each measurement needs to be compatible; (S,T,S) incompatibility means that *experiences* are representing non-related pieces of knowledge, i.e. they represent two different systems (conclusion 3);
- the *experiences* share only certain common *measurements*;
  - if the transformations are never equivalent, experiences represent two different pieces of knowledge or they are not consistent (conclusion 4); in order to verify which conclusion is the most appropriate, a further experience observing the Non-Common *Measurements* (NCM), i.e. the measurement that are not described by both the *experiences*, should be performed;
  - if the transformations are always equivalent then the experiences represent the same piece of knowledge (conclusion 5); moreover the NCM are not related to the other measurement in the experiences;
  - if the transformations are equivalent only for certain property values then experiences are equivalent only for those values of the NCM (conclusion 6).

To conclude, the algorithm performs a comparison between two *experiences* only on the basis of the (S, T, S)s: i.e. volumes, the evolutions of the volumes in the *time,* the *shapes* of the *measurements*. Each one of these comparisons is based on the proposed syntax. Since it is possible to perform this inference only on the basis of the syntax (i.e. the values of *tolerances, UOMs and range of values*) then the proposed algorithm does not instantiate abstract concept. Therefore it shows how the pieces of knowledge formalised with the proposed KR framework is not ambiguous. The presented algorithm is applied to a case study in the next section.

4. A CASE STUDY: HEAT TRANSFER IN A DEHUMIDIFYING WATER COIL

The subject of the application is a *water coil*, i.e. a family of components of the *fan coil* (Fig.7). A *water coil* is made of a set of punched aluminium fins. The fins are assembled on a coil of copper tubes that contains a flow of water. A motor activates a fan. The fan generates and orients an airstream towards the coil for the heat transfer. The usual *fan coil* implementation is for domestic uses.

Fig.7 - Pictures of the Trane fan coil.

The formalised knowledge about the *water coil* behaviour is from the following standards for the HVAC domain.

- The chapter 1 in (ASHRAE, 2009) about *psychometrics* has been used to model the air temperature and humidity relations.
- The chapter 4 about *heat transfer* in (ASHRAE, 2009) has been used to model the relations between heat transfer and fins geometry.
- The chapter 22 about *air-cooling and dehumidifying coils* in (ASHRAE, 2012) has been used to model the *water coil* system behaviour.
- The AHRI standard on *Forced-Circulation Air-Cooling and Air-Heating Coils* (Air-conditioning, Heating &

Refrigeration Institute, 2001) has been used to model the relations between the coil geometry and the heat transfer.

In the next sections, 1) the usage of the CAD software and 2) the formalisation of the knowledge about the *water coil* as instance of the *experience* concept are shown.

4.1. The water coil knowledge represented in CATIA V5

For the knowledge formalisation on the CAD environment CATIA V5, the *Knowledgeware*[2] module has been employed: this module has been used to express the constraints between the properties related to the volumes represented in the software. The instance of the conceptual model is represented as follows on CATIA.

An *experience* is a CAD model part or assembly model of parts. In the case of the *water coil* (Fig.8), the *experience* is about the coil assembly part that has as components the fins, the straight tubes and the *C-tubes* (i.e. short copper tubes that are bent and welded to close and orient the flow of water in the coil).

The *shape*, the *time* and the *space* of a *measurement* are represented as CATIA part parameters (at the left of the Fig.9). If the *measurement* is *T_A1* then *T_A1* represents the *shape*, *vol_T_A1* represents the *space* and *time_T_A1* represents the *time*. All the *times* are related each other's.

The parameter *vol_T_A1* and the other parameters representing the *spaces* of the *measurements* are related, by means of a CATIA *formula* (i.e. *vol_T_A1 = air_front\volume*), to the result of the *measure item* CATIA function (*air_front\volume* in the Fig.8). This type of relations allows to relate the *measurements* to the *spaces*.

For each *property*, the *UOM*, the *tolerance* and the *range of values* can be associated to the CATIA parameters.

The transformation are formalised using the *check* rules in the *knowledgeware* module of CATIA. For instance let us consider the following mathematical relation (i.e. an instance of *transformation*) between coil areas,

$$A_0=A_P+A_S \ .$$

The representation of this relation in CATIA is shown in Fig.10. In the same figure, a red or green light can be seen beside each constraint represented by a *check*. The green light notifies that the values of the parameters respect the constraint. The red light notifies that the constraint is violated. Notice that, as explained in section 3.5, a mathematical solver should verify if a set of values respect the constraints.

This usage of CATIA environment has allowed the representation of the *water coil* experience as in Fig.8 and Fig.9. In the next section, a simulation of how the *water coil* knowledge should be retrieved to perform a design stage is explained.

4.2. The design of the water coil

In this section a simulation of a design stage is considered: customer requirements are defined and then the retrieval of the *water coil* knowledge is shown. In order to start the design stage, the customer requirements should be formalised as *properties* of some *measurements*. In the proposed case study, the customer is represented by the following requirements (i.e. characterisation of **X**):

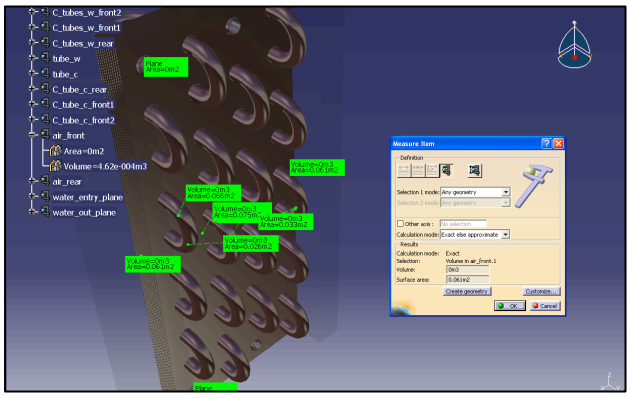

Fig.8 - The representation of the space of the measurement T_A1.

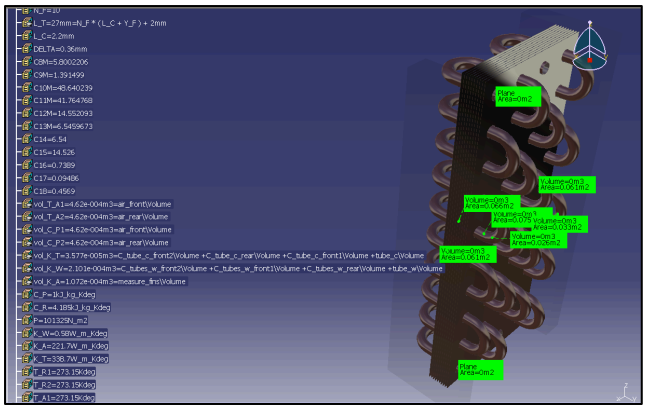

Fig.9 - The representation of the water coil experience on CATIA V5.

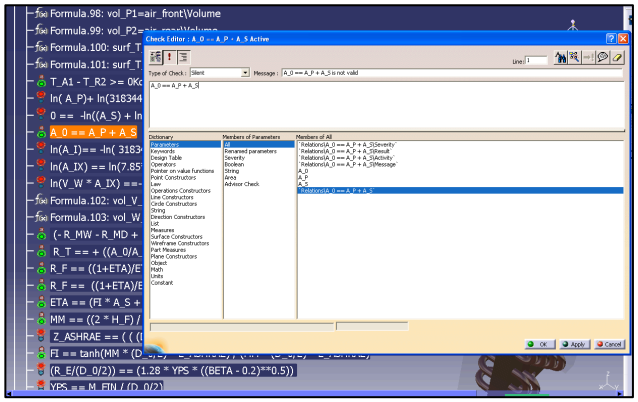

Fig.10 - The representation on CATIA of the transformations as check rules: beside each check, a red or green light shows if the parameter values are compliant with the formalised constraint.

- An air temperature and a humidity level have to be maintained in a certain volume.

[2] http://www.3ds.com/products-services/catia/portfolio/catia-v5/all-products/domain/Product_Synthesis/

- The temperature and the relative humidity represent two shapes. In order to characterise a *shape*, as for each *property*, the *UOM*, the *tolerance* and the *range of values* have to be specified. For the temperature (X_3) let us consider Celsius degrees, ±0.1 degrees as tolerance and 25.7 as value. For the relative humidity (X_4) consider % as UOM, ±0.01% as tolerance and 50% as value. The tolerance of the volume (*space*) characterisation is of ±0.1 mm. Since the values of the *shapes* have to be "maintained in a certain volume" there is no-dependence on time of the values of the *spaces* and *shapes*. The characterisation of the air is provided by the value of the specific heat in the same volume of the temperature and humidity *spaces*.

- The sensible power is constant.
  - The sensible power (X_5) characterises the required heat transfer. In the fan coil design this value characterises the room usage and features (e.g. presence of windows, computers, and ovens). In the proposed case the UOM is kW, the value is 6.9 kW with a tolerance of ±0.1 kW. The value is constant: therefore the reference time is the same of the temperature and humidity above referred. The space is equal to the volume of the above temperature and humidity.
- The temperature of a water flow is fixed at two points of the flow.
  - The water is characterised by its density and specific heat. The temperatures are: 1) 13.2 C° with a tolerance of ±0.1 (X_1); 2) 6.0 C° with a tolerance of ±0.1 (X_2). The value of the first temperature is equal or higher the one of the second temperature. The *spaces* of these measurements are not constrained by the customer.

All these *measurement properties* represent the customer requirements and they constrain the compatibility of an *experience*. Each property of the measurement that is not specified is considered as an engineer degree of freedom. In order words, the defined properties *ranges*, *UOMs* and *tolerances* are the only limits to determine the compatibility of a formalised experience. All other (S, T, S)s defined in the experience are considered as not relevant for the customer satisfaction. Therefore their values can be fixed on the basis of the impacts on the manufacturing costs.

In the next subsections, the simulation of the knowledge retrieval to perform a design stage is discussed. Starting from the above requirements, the *experience*s retrieval will allow the connection of the customer requirements to the degrees of freedom that the engineers can handle to meet these latter.

Customer-to-Product

In the above requirements a volume with specific heat, temperature, humidity and sensible power has been specified. The *time* of all these measurements is the same, i.e. stationary conditions holds. In all the eventual available *experiences*, a volume in which all these measurements are observed is searched. Therefore the association of the customer-defined *shapes* with the ones in the *experiences* is based on the *shapes* that are measured in a certain volume at a given time.

In order to verify that a customer-defined *shape* matches with a *shape* in an *experience*, the *UOMs* (values of strings have to match), the *ranges of values* (customer-defined values need to be included in the ranges of the *experience*) and the *tolerances* (customer-defined values need to be equal or higher than values in the *experience*) of the *properties* of the *measurements* need to match.

In the volume in Fig.11, the *water coil experience* represents three measurements:

- 15 to 30 ±0.01 °C;
- 35 to 65 ±0.01 %;
- 0.5 to 13 ±0.1 kW.

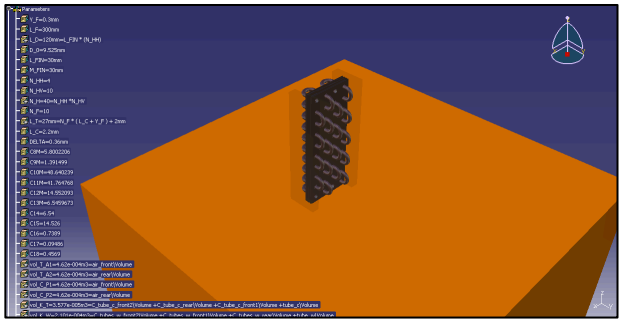

Fig.11 - The coil and the volume of air in the customer requirements.

Also for the water temperatures the reasoning is the same. Two different temperatures in two different surfaces are measured on a flow stream of water at the coil entry and at the coil exit. The density and the specific heat measured in the surfaces matches with the customer-defined ones. The temperature *shapes* observed in the two couples of surfaces (in orange, at the top and at the bottom of Fig.12) in the experience are:

- 5 to 12 ±0.01 °C;
- 10 to 20 ±0.01 °C.

Since the five *measurements* (3 temperatures, humidity and power) respect the constraints about the *UOMs*, *ranges* and *tolerance* of the customer-defined (S, T, S)s, the *water coil experience* formalises the knowledge to meet the requirements of the defined customer.

Notice that in this case study, the customer defined *spaces* have not been constrained. If a precise region of the space is constrained by the customer, a volume comparison is also required to verify the compatibility of an experience with the customer requirements. This verification has to be performed on the basis of the UOMs, ranges of values and tolerances of the customer defined *spaces* with the related region defined in the experience.

Since the *spaces* are surfaces or volumes in the CAD model, two regions of the space are equivalent when their intersection is equal to both the regions. In order to check the compatibility of the *spaces*, this verification should be performed for each value of each properties relative to each volume to be compared.

Product-to-Process

Once the water coil experience can meet the customer requirements, the measurements that match with the customer-defined properties are removed from the engineer's degrees of freedom. Indeed, all the non-constrained

properties (i.e. **Y**) are de facto the properties that the engineer should control to meet the customer requirements.
In the water coil examples, the degrees of freedom include the thermal resistance of the tube and fin material, the water mass flow, the exiting air temperature and so on. Also the geometries of the measurement spaces are included. All these parameters become the requirement of a manufacturing or assembly process.

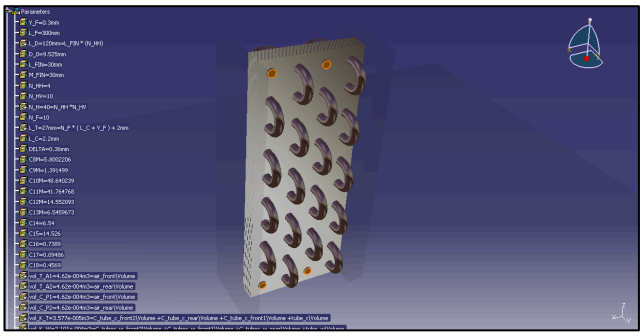

Fig.12 - Surfaces in which the entering and exiting water temperature can be observed.

As for the customer-defined requirements, the compatibility is based on the *UOMs*, the *ranges of values* and the *tolerances*. As general rule, the starting point for the compatibility verification is the volume comparison between the requirements (depending on customers, on products or on other processes) and the volume evolutions in the *experiences* representing the processes. In other words, all the requirements have to match with one or more experiences at least at a given time (to guarantee that the process is able to build what is required to meet the requirements).
When an assembly process is formalised, at the starting point (at *time=0*), the CATIA environment should check the components that are still not connected. Let us consider the transformations at that time. A subset of *properties* is identifiable: each *property* belongs only to a subset and can be related only to *properties* of the same subset. Each subset can be considered as an independent component. Therefore the requirements of the assembly *experience* can be met by more than one other *experience* (e.g. one for each independent component). Since the identification of the subset should add another complexity to the knowledge retrieval, an equivalent solution is deployable, i.e. add an ID of the component in the name of the related properties (e.g. the parameter *Y* should become *Y_x*, where *x* is the ID of the component).
In the water coil case, the experiences that are retrieved to connect the customer requirements to the manufacturing process properties are the following. The highlighted *Ys* represents the characterisation of **Y**.

- The installation process: since no-effects of this process on the heat transfer are modelled in the considered HVAC standards, the installation is considered as a movement of the water into the coil and of the coil in the airstream. In the CATIA environment this time-dependences of the parameters have been modelled by means of the macros (Fig.13). The relative position of the coil and the airstreams depend on the value of the parameter *time_1*.
    - Here the airstream speed is considered equal to the fan speed (*Y_11*), i.e. 3.56±0.01 m/s.

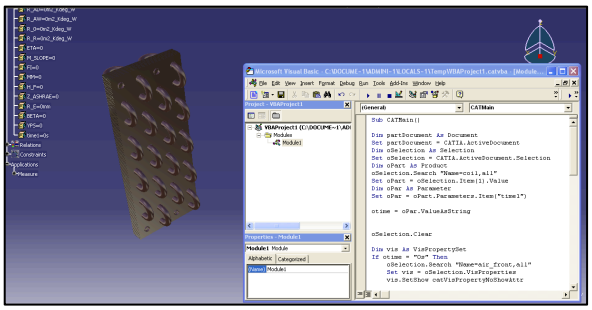

Fig.13 - The macro for representing the time-dependences in the installation process.

- The C-tubes welding process: the welding process has been considered as a simple movement of the C-tubes. The position of the C-tubes determines the number of coil circuits (*Y_2*) that is equal to 4 circuits.
- The tube expansion: this process is modelled as a change of the relative position of the tubes and the fin holes.
- The assembly stage: the assembly process is modelled as a change in the relative positions of the aluminium fins and the copper tubes.
- The fin punching process: the process is modelled as the cutting (i.e. change of the length of the fin) and punching (i.e. manufacturing of the fin holes) of an aluminium sheet. The degrees of freedom of this process are as follows considered as characterising the process definition:
    - number of horizontal holes (Y_1), with value equal to 4;
    - the punching tool depth that impact the fin collar (Y_8) that determine the distance between two fins, equal to 2.2±0.01 mm ;
    - the aluminium sheet height (Y_7), equal to 400±0.01 mm;
    - the aluminium sheet thickness (Y_5), equal to 0.15±0.01 mm;
    - the horizontal (Y_9, 25.4±0.01 mm) and vertical (Y_10, 22±0.01 mm) distance between two punching tool features (i.e. the features that hole the aluminium sheet).
- The tube cutting process: the process is modelled as the change of length of a copper tube. The degrees of freedom of this process are considered as characterising the process definition:
    - the copper tube thickness (*Y_3*) is 0.81±0.01 mm;
    - the copper tube diameter (*Y_4*) is 10.06±0.01 mm;
    - the frequency of tube cuts that impacts the tube length (*Y_6*); the resulting length is equal to 330±0.01 mm.

4.3. A mathematical model to test the applicability of the retrieved knowledge for the customer

The ranges of values for the customer (the *X* variables, **X**) and the process (the *Y* variables, **X**) *properties* and the *tolerances* allow to build the ranges for each customer or process in the MINLP model. The model have been formalised and solved in Lingo[3].

A small number has to be defined the range in Lingo for each property. Indeed, the solver does not accept "<" and ">" constraints. Therefore in order to formalise the value of a property for a feasibility test, a small number is needed.

For instance, let us consider the temperature of the water exiting the coil (i.e. the customer requirement *X_1*). The constraint that has to be represented in Lingo for the customer *X_1=13.2±0.1* (i.e. *13.1 < X_1 < 13.3*) is the following

$$13.11 \leq X_1 \leq 13.29,$$

where the lower bound is equal to

13.2 (nominal value) - 0.1 (customer tolerance) + 0.01 (knowledge precision)

and the upper bound is equal to

13.2 (nominal value) + 0.1 (customer tolerance) - 0.01 (knowledge precision).

Since for the processes there is only the knowledge precision, the process properties ranges need the definition of another small number to build the lower and upper bounds. For instance, consider the external coil tube diameter (Y_4). The lower bound is calculated as follows

10.06 (nominal value) - 0.01 (knowledge precision) + 0.001 (number smaller than precision),

instead the upper bound is equal to

10.06 (nominal value) + 0.01 (knowledge precision) - 0.001 (number smaller than precision).

The properties can be also represented as discrete intervals (e.g. Y_1). In this case the constraint in Lingo is a simple equality, e.g. *Y_1=4*.

Once the all the ranges for the properties are formalised in Lingo, a feasibility test can be performed to check if a solution that respect all the constraint exists. The eventual solution represents the values of the process properties to manufacture a product that is able to satisfy the customer requirements.

At the left of Fig. 14, a screenshot of one MINLP model on Lingo is shown. At the top of the model there is the definitions of the ranges for the customer-process couple. At the bottom all the transformations in the retrieved experiences are formalised in Lingo language to represent the constraints of the model.

The execution time of the solver is limited to 10s in the case showed. At the right of Fig. 14, the report of the test with the values above is shown.

4.4. The unambiguity of the water coil knowledge

In order to test the conclusions (1 to 6) of the algorithm proposed (i.e. the algorithm about the knowledge mapping between two *experiences*), examples of experiences about the water coil are considered.

The algorithm proposed in section 3.5 is expected to estimate the relations between couples of experiences. The algorithm can infer about:

- the *inconsistency* of the two *experiences*, i.e. the *experiences* formalise the same *measurements* but they have two different behaviour formalised by the *transformations* (conclusion 1);
- the complete or partial equivalence of the represented experiences (conclusions 2, 5 and 6);
- the absence of relations between the two experiences (conclusion 3, 4).

Two main conditions determine the difference of conclusions that the algorithm can provide: 1) the verifications of the compatibility of the measurements that are based on the (S,T,S)s comparison, as for the knowledge retrieval in the design stage; if two measurement from the two experiences are compatible only for part of the ranges of values, the analysis of the algorithm is performed only for these values; 2) the verifications of the transformations that are realised by comparing the constraints in the two experiences that are violated by a set of values for the common *measurements*. The first kind of comparisons can be performed on the basis of the parameter values of the *experiences* formalised on CATIA model. The second kind of comparison should be performed on a mathematical solver able to deal with MINLP models, e.g. Lingo. In the following subsection, all the possible conclusions are discussed on the *water coil* example.

The two experiences represent the same measurements.

The branch of the flowchart that deals with two experiences that represent the same measurements is reported at the left of Fig.6. In this case, the two experiences formalise an experiment done on the same measurements (i.e. (S, T, S)s are compatible).

Let us consider the *water coil* example. Suppose that the same measurements represented in the above shown experience are formalised in two different CATIA models. In order to test the constraint violation, set of values for the observed measurements have to be tested on the solver.

Conclusion 1 - When the solver results are not compatible for the two experiences, they are inconsistent. Two are the possible explanations for this result. 1) The inconsistency is due to a factor that is not observed in both the experiences and that takes different values during the two experiences, e.g. the thermal resistance of the duct that orient the airstream. 2) The inconsistency is due to a high level of tolerance for some properties (e.g. rugosity if the inner tube surface) that can hide effects of the (S, T, S)s on the system behaviour. In both of these situations, the mathematical regression (based on data of the *experience*-related observation) can lead to different results and so to different mathematical relations between the (S, T, S)s, i.e. the transformations. Therefore the Lingo solver launched on two models that have the same values for (S, T, S)s but different constraints (i.e. transformations in the mathematical model) should provide different results, e.g. one model is feasible and the other one is infeasible for the same values of the properties.

[3] http://www.lindo.com/index.php?option=com_content&view=article&id=2&Itemid=10

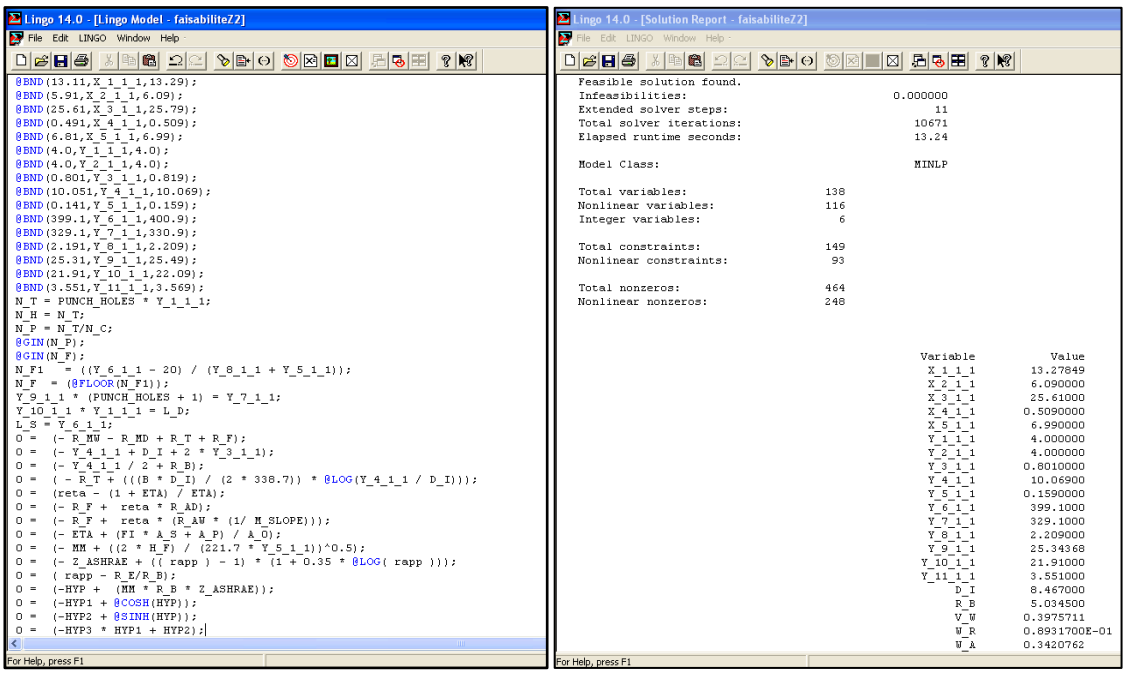

Fig. 14 – Screenshot of the solver Lingo.

Conclusion 2 - When the solver gives the same result (feasible or not) for the two experiences for each given set of values for the (S, T, S)s, the two experiences are completely equivalent. In the case of the water coil, this means that two experiences on CATIA describe the same system (same (S, T, S)s and equivalent transformations) therefore they can be used as alternative pieces of knowledge in a design stage.

The *conclusion 3* is not explained here because it is simply due to an incompatibility of the (S, T, S)s. This means that the experiences are representing a completely different piece of knowledge, i.e. the non-ambiguity can be trivially verified. ***The two experiences share the representation of some measurements***.

The two *experiences* can eventually share only part of the observed measurements (at the right of the flowchart in Fig.6). In this case the verification of the constraint violation have to be done with a particular attention on the values considered for the non-common *measurements* (NCM).

Conclusion 4 - If the experiences (S, T, S)s are never compatible then one of the two situation holds: 1) they are describing different pieces of knowledge or 2) they are inconsistent. In order to verify what is the right conclusion, further observations considering the NCM should be performed. This means that for no-values of the thermal resistance of the duct, the solver gives compatible results between the two experiences.

Conclusion 5 - If for all values of the NCM, the solver results are compatible for values of the common (S, T, S)s, the NCM are not influential on the behaviour of the represented system. This means that the even if the thermal resistance of the duct is considered in only one experience, its values cannot impact the relations between the other *measurements*.

Conclusion 6 - When the solver results are compatible only for some values of the NCM that values are the values that would have been observed in the experience that does not represent them if these NCM were taken into account. This means that, when the solver gives compatible results, the related values of the thermal resistance of the duct would have been observed during the observation run following the knowledge formalised in the *experience* that does not represent it.

5. DISCUSSION AND CONCLUSIONS: LIMITS AND THE AUTOMATIC MAPPING PERSPECTIVES

Conclusions

This paper should be seen as an attempt to provide an alternative to logics for design knowledge representation (KR). The paper deals with the limits highlighted during the works presented in (Giovannini, Aubry, Panetto, Dassisti, & El Haouzi, 2012) and details the proposal in (Giovannini, 2015). Due to the generality of the proposal, this paper represent only a first step ("explorative") for the formulation of a consolidated approach.

The main point of the paper's contribution is about an approach general enough to provide a knowledge model that satisfies statement S1. To do so, the concept of *measurement* has been developed to provide a framework for setting *unambiguous* KR. The *measurement* concept — from the transposition of perception in the *anti-logicist* corpus of knowledge — allows to formalise the knowledge avoiding the personal interpretation of the user. In this way, it is possible to guarantee a univocal use of the represented knowledge, i.e. the use intended by the knowledge modeller.

In order to support the expressed positions, authors have developed an algorithm to demonstrate the KR *unambiguity* from the proposed framework. This algorithm has been applied on an industrial case. The aim of the authors was to show how the mapping process between two pieces of knowledge is based only on the syntax adopted, thus avoiding

the interpretation of concept described by means of the natural language.

A method to instantiate the presented framework and enrich with knowledge the CAD model has been proposed. This method has been implemented in CATIA platform but it is theoretically applicable on every parametric CAD application. This method can support the design for variety approaches by providing an unambiguous way to connect heterogeneous pieces of knowledge.

Limits

The limits of the proposed solution concern mainly the complexity of the represented knowledge and thus of the feasibility test. The proposed framework provides an unambiguous KR working on the syntax for the knowledge retrieval and mapping. But the syntax is based on set of values for the properties. And the more the complexity grows (i.e. number of properties and constraints to represent **X-Y** relations) the more the knowledge becomes hard to manage, for the knowledge retrieval, the mapping and for the solution of the MINLP problems. Indeed, even for the water coil, that is only a component, the time to perform a run of the algorithm takes about 10 seconds. For more complex components and systems, coping with the tests with an exact algorithm may become untreatable.

Perspectives

As shown with the algorithm to demonstrate the unambiguity of the proposed KR, a knowledge mapping and retrieval is possible only as a function of the syntax of the representation. The main perspective related to the proposed knowledge representation framework is to deeply understand the implication of an unambiguous knowledge representation on problems like the automatic mapping of knowledge. Indeed, the mapping of knowledge is at the present semi-automatic (Kalfoglou & Schorlemmer, 2003; Rahm & Bernstein, 2001). This conclusion can lead to some relevant improvements to the automation of the design knowledge reusability. Moreover, improvements in mapping can also impact the interoperability about the product and process information (Panetto, Dassisti, & Tursi, 2012).

A connected perspective concerns the automation and improvement of the knowledge retrieval as formalised in section 4, i.e. on a CAD environment. Since this processes are based also on the volume comparison, an appropriate algorithm can provide a more effective support to an automatic mapping or knowledge retrieval (Shah, Anderson, Kim, & Joshi, 2001).

6. REFERENCES

Agre, P. E., & Chapman, D. (1987). Pengi: An Implementation of a Theory of Activity. In *Association for the Advancement of Artificial Intelligence* (Vol. 87, pp. 286–272).

Air-conditioning, Heating & Refrigeration Institute. (2001). AHRI Standard 410-2001 with Addenda 1, 2 and 3: Forced-Circulation Air-Cooling and Air-Heating Coils.

ASHRAE. (2009). *2009 ASHRAE Handbook: Fundamentals*. American Society of Heating, Refrigerating & Air-Conditioning Engineers, Incorporated.

ASHRAE. (2012). *2012 ASHRAE Handbook: heating, ventilating, and air-conditioning systems and equipment*. Atlanta, Ga.: American Society of Heating, Refrigerating & Air-Conditioning Engineers, Incorporated.

Baader, F., Horrocks, I., & Sattler, U. (2008). Description logics. *Handbook of Knowledge Representation*, *3*, 135–179.

Brewka, G., Niemelä, I., & Truszczynski, M. (2007). Nonmonotonic reasoning. *Handbook of Knowledge Representation*, 239–284.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, *47*(1), 139–159.

Chauvet, G. A. (1993). Hierarchical functional organization of formal biological systems: a dynamical approach. I. The increase of complexity by self-association increases the domain of stability of a biological system. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *339*(1290), 425–444.

Darwiche, A. (2008). Bayesian networks. *Handbook of Knowledge Representation*, *3*, 467–509.

Davis, E. (1998). Naive physics perplex. *AI Magazine*, *19*(4), 51.

Forbus, K. D. (2008). Qualitative modeling. *Handbook of Knowledge Representation*, *3*, 361–393.

Gelfond, M. (2008). Answer sets. *Handbook of Knowledge Representation*, *1*, 285.

Giovannini, A. (2015). *A knowledge representation framework for the design and the evaluation of a product variety*. PhD Thesis - University of Lorraine, France.

Giovannini, A., Aubry, A., Panetto, H., Dassisti, M., & El Haouzi, H. (2012). Ontology-Based System for supporting Manufacturing Sustainability. *Annual Reviews in Control*, *36*(2), 309–317.

Giovannini, A., Aubry, A., Panetto, H., El Haouzi, H., Pierrel, L., & Dassisti, M. (2014). Approach for the rationalisation of product lines variety. In *Proceedings of 19th IFAC World Congress* (Vol. 19(1), pp. 3280–3291). Cape Town, South Africa.

Gomes, C. P., Kautz, H., Sabharwal, A., & Selman, B. (2008). Satisfiability solvers. *Handbook of Knowledge Representation*, *3*, 89–134.

Grosof, B. N., Horrocks, I., Volz, R., & Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web* (pp. 48–57). ACM.

ISO/IEC. (2007). ISO/IEC 24707 Information technology–Common Logic (CL)–A Framework for a Family of Logic-Based Languages.

Kaelbling, L. P. (1987). An architecture for intelligent reactive systems. *Reasoning about Actions and Plans*, 395–410.

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: the state of the art. *The Knowledge Engineering Review*, *18*(01), 1–31.

Lieber, R., Dupont, J.-M., Bouffaron, F., & Morel, G. (2013). Improving physical-physiological interaction requirements for maintenance enabling systems specification. In *12th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*. Las Vegas, États-Unis.

Lifschitz, V., Morgenstern, L., & Plaisted, D. (2008). Knowledge representation and classical logic. *Handbook of Knowledge Representation*, *1*, 3–88.

Panetto, H., Dassisti, M., & Tursi, A. (2012). ONTO-PDM: product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. *Advanced Engineering Informatics*.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, *10*(4), 334–350.

Rossi, F., Van Beek, P., & Walsh, T. (2008). Constraint programming. *Handbook of Knowledge Representation*, *1*, 181.

Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th international joint conference on Artificial intelligence* (Vol. 1, pp. 466–471).

Shah, J. J., Anderson, D., Kim, Y. S., & Joshi, S. (2001). A discourse on geometric feature recognition from CAD models. *Journal of Computing and Information Science in Engineering*, *1*(1), 41–51.

Sowa, J. F. (2008). Conceptual graphs. *Foundations of Artificial Intelligence*, *3*, 213–237.

Struss, P. (2008). Model-based problem solving. *Handbook of Knowledge Representation*, 395.

Van Heijenoort, J. (1977). *From Frege to Gödel: a source book in mathematical logic, 1879-1931* (Vol. 9). Harvard University Press.

Whitman, L. E., & Panetto, H. (2006). The missing link: Culture and language barriers to interoperability. *Annual Reviews in Control*, *30*(2), 233–241.