

Chapter #

FOUNDATIONS FOR A CORE ONTOLOGY OF MANUFACTURING

Stefano Borgo ¹ and Paulo Leitão ²

¹*Laboratory for Applied Ontology, ISTC-CNR, via Solteri 38,38100 Trento, Italy, borgo@loa-cnr.it;* ²*Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal, pleitao@ipb.pt*

Abstract: An initial fragment of a core ontology for the manufacturing domain is presented and motivated. It consists of an ontological classification of ADACOR concepts according to the DOLCE foundational ontology. The ontology is conceptually transparent and semantically explicit thus suitable for information communication, sharing, and retrieval. The system here described considers entities performing the manufacturing scheduling and control operations only.

Key words: Foundational Ontologies, Manufacturing Control Systems.

1. INTRODUCTION

If “communication age” and “information era” are popular terms highlighting the characteristics of the time we are living, the struggle for mutual and reliable understanding that is nowadays recognizable across all application domains suggests that a more appropriate term to capture the trend in today and the near future research could be “the semantic period”.

This special nature of our time can be seen in application domains as well and the popularity of the term “ontology” in manufacturing is an example. Generally speaking, this term refers to knowledge engineering artifacts that are constituted by a natural or formal language plus a set of assumptions and constraints (Guarino, 1998).

The development of an ontology may take from a few hours up to months or even years depending on the choice of the language, the covered topics, and the level of formality and precision. The ultimate goal is the unambiguous description of a certain “reality” of interest. The type of application or the sought generality leads to the choice of a construction methodology which, in turn, guarantees the reliability of the resulting ontology. There are several possible ways to evaluate an ontology among which the expressivity of the adopted language (glossaries, controlled/natural/formal languages), the purpose of the ontology (knowledge sharing, domain modeling, information retrieval, natural language processing,...), the domain covered (management, business, law, medicine, digital libraries...), the structural complexity of the system

(tangledness of the taxonomy, degree of branching, depth of the hierarchy, modularity,...) and so on.

Taking semantics at face value, one can roughly divide ontological systems in classes starting from those with weak semantics like glossaries, thesauri, and taxonomies (*terminological ontologies*) and ending with rich logical theories (*formal ontologies*). The first type of ontologies, like WordNet (Fellbaum, 1998), is helpful in organizing catalogs, databases and protocols where only terminological services are needed. When looking for conceptually transparent and semantically robust systems, sophisticated knowledge structures like formal ontologies need to be used. Formal ontologies can be very general (*foundational*) or domain dependent (*core*). Generally, a core ontology is aligned to a foundational one to guarantee interoperability in open and evolving environments.

Nowadays only a few projects have produced widespread formal ontologies, e.g. the GALEN project¹ in the medical field. There are several reasons for this. Formal ontologies are relatively new and only in the last few years reliable methodologies have been introduced and consistently applied (Oltamari et al., 2002). Moreover, the development of applications based on these ontologies is sometimes demanding so that often research concentrates on smaller projects, e.g. (Bertolazzi et al., 2001), whose results are unfortunately hardly generalizable.

This paper starts from a widely used foundational ontology and develops a formal ontology for manufacturing scheduling and control environments.

¹ <http://www.opengalen.org/>

The aim is to extend this approach the other aspects of the manufacturing domain in order to build a core ontology for this domain that guarantees: (1) integration with a well organized and accepted foundational ontology; (2) accessibility to agents in the manufacturing domain; and (3) suitability for product and process modeling as well as for information sharing, exchange and retrieval.

It is important to understand that the adoption of foundational ontologies does not force to change the production nor the production organization. If ontological considerations will likely suggest changes in the overall enterprise information system to optimize knowledge management and to guarantee data consistency, the very fact that a foundational ontology focuses on data *content* and *description* should make clear that the production and its organization do not concern the ontology itself. Indeed, foundational ontologies are independent from the type of products or the number of their variants, their functionalities, qualities or else. Actually, foundational ontologies can improve the product quality by giving a tool to handle non-functional requirements (security, reliability) which are hard to consider within standard architecture languages.

This paper is organized as follows. In section 2 the application domain is introduced and specific concepts are highlighted. Section 3 gives a general overview of important projects that bring the ontological perspective into the manufacturing area. The next section describes the chosen domain architecture and, in section 5, the DOLCE foundational ontology is presented. The core of the paper is section 6 where the alignment between

the domain architecture and the formal ontology is motivated and carried out. A few examples show how to express relevant information in the resulting formal language. The last section briefly discusses the use of the proposed manufacturing ontology in applications, adds some final considerations, and lists future steps to be carried out.

2. SCHEDULING AND CONTROL PROBLEM

DESCRIPTION

The manufacturing domain is and will be in the future one of the main wealth generators in Europe. Nowadays it represents approximately 22% of the GNP of the EU and 70% of the employment (European Commission, 2004). The development of adaptive, digital, networked and knowledge-based manufacturing processes is the key factor for the competitiveness and success of a manufacturing enterprise.

This study applies to manufacturing scheduling and control systems: it considers a manufacturing enterprise that produces discrete items and models components of the factory plant as well as aspects of the scheduling, monitoring, and execution processes.

2.1 Manufacturing System Description

A manufacturing enterprise *produces* products which are offered to the market. Within the enterprise, the products are described by the *product model*, which contains all technical data and describes the structure of a

product (list of sub-products or parts that assembled constitutes the product), and by the *process model*, which defines how to produce the product.

The process model specifies the *process plan*, that is, a list of operations necessary to produce a part. In this context, an *operation* is a job to be executed in order to produce the product and is characterized by a set of information, such as estimated processing time, description, precedence, and requirements. Assembly, storage, transportation, manipulation, maintenance and inspection, are examples of operations.

A customer interacts with a company to order one of the available products or a new product. This order, known as *customer order*, must include reference to a product, a quantity, a deliver date, and a price. Additionally, the enterprise management system creates *forecast orders* to anticipate the market demands. The manufacturing planning convert the customer and forecast orders into *production orders*, aggregating if possible several customer and forecast orders into a production order, to obtain volume and transport advantages. A production order is indexed to a product object and comprises a list of work orders. A *work order* is the description of an operation (a job) and thus is a part of a process plan. Work orders are intended to be executed by *resources* such as movers, transporters, drilling machines, milling machines, turning machines and tools. Each resource is an entity that can execute a certain range of jobs, when it is available, as long as its capacity is not exceeded.

The *shop floor* consists of a group of resources with different characteristics (spindle speed, list of tools and grippers, payload, time

autonomy, work volume, repeatability, etc.), whose combined features allow to execute the products. The *factory model* describes each individual shop floor's resource as well the logical and physical organization of these resources. The availability of a resource is represented by an *agenda* that indicates the list of work orders allocated to the resource over the time. In particular, the agenda comprises time slots where the resource is: free, allocated to execute orders, temporarily out of service (e.g. due to maintenance) and out of service (e.g. due to a break in the provision of needed elements like water or electric power).

2.2 Manufacturing Control Description

The main functions that a *manufacturing control system* (*MCS* for short) fulfills are process planning, resource allocation planning (scheduling), plan execution, and pathological state handling.

The production of a product involves the execution (according to a precedence diagram) of the steps defined in the process plan. At the *process planning* level, the *MCS* launches the production orders to the shop floor together with a process plan. The latter provides the required sequence of operations and the required machine type for each operation. Based on the available resources, it is possible to create alternative process sequences (aiming to achieve flexibility), each one indicating the exact resource that should execute each operation.

Through the *resource allocation planning*, the *MCS* schedules the necessary operations to produce the parts (including processing, transport,

maintenance and set-up operations) taking into account the process plans, the constraints and resources capacity. The goal is to produce the products while minimizing the costs and increasing the productivity. Also, at this level the *MCS* considers possible reorganizations of the production unit (in general by varying the resource allocations) if a modification in demand or machine failure makes it necessary.

The *plan execution functions* of the *MCS* take care of the physical implementation of the schedule into the factory. The scheduled orders are dispatched to the manufacturing plant, i.e. the resources, and a monitoring activity of the production progress takes place. The reaction to disturbances is first considered by the *MCS* at the level of the plan execution but may imply re-scheduling of the operations to minimize the effects of the disturbance and, in some cases, the interruption of the production process.

The *pathological state handling* level intends to keep the system in a safe state, in order to avoid and/or recover from undesirable system states, such as deadlock.

3. ONTOLOGIES AND THE MANUFACTURING DOMAIN

In order to improve agility and flexibility, nowadays one uses distributed approaches in developing manufacturing control applications. These are built upon autonomous and cooperative entities, such as those based on

multi-agent and holonic systems. Holonic Manufacturing System (HMS)² translates to the manufacturing world the concepts developed by Arthur Koestler for living organisms and social organizations (Koestler, 1969). Holonic manufacturing is characterized by holarchies of holons (i.e., autonomous and cooperative entities), which represent the entire range of manufacturing entities. A holon is a part of a (manufacturing) system that has a unique identifier, may be made up of subordinate parts and, in turn, can be part of a larger whole.

3.1 Manufacturing Interoperability

In distributed manufacturing environments (with autonomous entities representing machines, cells, factories or even enterprises) it is important to guarantee the compatibility between the distributed entities or applications (i.e. issues related to interfaces and protocols) and to verify that the semantic content is preserved during the exchange of messages between distributed entities. Thus, interoperability in distributed platforms increases the need for shared ontologies. Specifically, the term ‘manufacturing interoperability’ is related to the ability to share technical and business information throughout a distributed factory plant or even an extended or virtual manufacturing enterprise. A study commissioned by NIST (National Institute of Standards and Technology) (Brunnermeier and Martin, 1999) reported that the U.S. automotive sector alone expends one billion dollars per year to solve interoperability problems.

² <http://hms.ifw.uni-hannover.de/>

The ontologies currently used in the manufacturing domain are the result of non-coordinated efforts and relinquish the interoperability with other agents communities. Indeed, proprietary manufacturing ontologies have been developed to support the interoperability between distributed entities belonging to the same platform only. The lack of interoperability between different agent-based or holonic manufacturing control platforms pushes for a common manufacturing ontology capable of merging (or at least of communicating adequately with) these.

3.2 Toward Standard Manufacturing Ontologies

Since interoperability has become a central issue in the manufacturing domain, several efforts have been undertaken to develop standard mechanisms for the unambiguous exchange of information in this area. This section reviews some of these efforts and highlights important aspects for a general manufacturing ontology.

The EDI (Electronic Data Interchange) is a standard suitable for applications that want to exchange data through standard formats. EDI is limited to business data only, thus those applications that need to manage engineering and technological information have to resort to other standards targeting more closely the exchange of product data.

Several proposals have been presented for this goal. The IGES (Initial Graphics Exchange Specification) and SET (Standard d'change et de Transfert) have been important stimuli for the data exchange standardization but they fall short of solving the entire problem because the proposed

standardization considers the information at the geometrical level and disregards the technological data. STEP, Standard for the Exchange of Product Model Data developed by the International Organization for Standardization (ISO), defines a standard data format for exchanging a complete product specification (e.g. geometry and production process) between heterogeneous CAD/CAM systems or entities belonging to a supply chain. ISO developed also Plib, Parts Library (<http://www.tc184-sc4.org/>), which is a computer-interpretable representation of parts library data to enable a full digital information exchange between suppliers and users. Plib and STEP share a common technology basis and are completely interoperable, one focusing on product data, the other on libraries of parts. However, since STEP refers to the product information only and Plib to representation and exchange of part library data, the process and enterprise engineering information are out of their scope.

Another set of initiatives seek to fulfill the gaps. The Process Specification Language project (PSL) (Schlenoff et al., 1996) aims to develop a general ontology for representing manufacturing processes to serve as an interlingua to integrate multiple process-related applications throughout the manufacturing life cycle. A Language for Process Specification (ALPS) (Catron and Ray, 1991) identifies information models to facilitate process specification and to transfer this information to process control. The TOVE, Toronto Virtual Enterprise project (Fadel et al., 1994), defines a domain-specific formal ontology for enterprise modeling which is not connected to foundational ontologies. The Enterprise Ontology provides

“a collection of terms and definitions relevant to business enterprises to enable coping with a fast changing environment through improved business planning, greater flexibility, more effective communication and integration” (Uschold et al., 1998). The goal of the Process Interchange Format project (PIF) (Lee et al., 1998) is to support the exchange of business process models across different formats and schemas. Finally, the Plinius project (van der Vet et al., 1994) aims to define a domain-specific ontology for mechanical properties of ceramic material.

In spite of the referred efforts to develop ontologies in areas related to manufacturing, as of today no formal ontology is available in the manufacturing domain. Nonetheless, it is recognized that the application of formal ontologies to support the interoperability between agent-based and holonic manufacturing control applications could provide a reliable and durable solution to this problem. The ongoing activity of the holonic manufacturing community within FIPA (Foundation for Intelligent Physical Agents) to adequate the FIPA specifications to the manufacturing requirements would benefit as well from the adoption of well-justified and organized formal ontologies, that is, ontologies furnished with a deep logical characterization.

3.3 Foundational Ontologies

As anticipated in section 1, foundational ontologies are formal ontologies devoted to facilitate mutual understanding in the large and are developed independently of specific domains. Because of this approach, they comprise

only general concepts and relations, and to be applied they need to be populated with notions specific to the domain of interest. Indeed, these ontologies aim at setting a general framework that can be tailored to any application domain; in this way they furnish a reliable tool for information sharing and exchange in all areas. In short, foundational ontologies are characterized by the following crucial properties: they are *general* in the sense that they limit themselves to the most reusable and widely applicable concepts leaving to the user the population of the ontology with more specific concepts; they are *reliable* since they are logical theories with rich axiomatizations and with careful analysis of their formal consequences (theorems); and they are *well organized* because the construction of a foundational ontology is based on philosophical principles whose choice is explicitly motivated.

Just a few foundational ontologies have been developed to a satisfactory level in the literature: DOLCE, the Descriptive Ontology for Linguistic and Cognitive Engineering (Masolo et al., 2003) <http://www.loa-cnr.it/DOLCE.html>; GFO, the General Formal Ontology (Heller and Herre, 2003) <http://www.onto-med.de>; OCHRE, the Object-Centered High-level Reference Ontology (Masolo et al., 2003); and, although still in a preliminary form, BFO, the Basic Formal Ontology (Masolo et al., 2003), <http://www.ifomis.de>. Two other systems are sometimes considered in the literature although they are not, strictly speaking, foundational ontologies. These are OPENCYC (<http://www.opencyc.com>) and SUMO (<http://www.ontologyportal.org>).

Two issues should be carefully analyzed in choosing a foundational ontology for applications: the ontology must include a set of conceptual distinctions sufficient for that domain (e.g. the distinction between abstract and concrete, agentive and non-agentive, etc.), and all the relevant entities should be clearly characterizable within the ontology (e.g. orders, resources, sensors, measurable qualities, etc.)

4. THE ADACOR MANUFACTURING ONTOLOGY

In the manufacturing domain, manufacturing control approaches are implemented and improved continuously. To ground the discussion, this paper selects one architecture and provides an ontological assessment of its concepts. That is, the notions of this architecture are analyzed for their ontological commitment (Guarino, 1998), classified following a foundational ontology, and formalized accordingly. The result of this process is a formal system comprising the architecture notions and their ontological organization, that is, a system that can be taken as a *core ontology* for the manufacturing domain. This system is not limited to the chosen architecture. New concepts can be added from other architectures by following the methodology (see section 6) and other systems can openly and safely communicate with any aligned architecture (if they can manage the language of the ontology) even without being aligned themselves.

ADACOR (ADaptive holonic COntrol aRchitecture for distributed manufacturing systems) (Leitão et al., 2005) is the architecture here

analyzed. Based in the HMS paradigm, it addresses the agile reaction to disturbances at the shop floor level in volatile environments and it is built upon a set of autonomous and cooperative holons, each one being a representation of a manufacturing component, i.e., a physical resource (numerical control machines, robots, etc.) or a logic entity (orders, etc.).

ADACOR defines its own proprietary manufacturing ontology, expressed in an object-oriented frame-based manner as recommended in the FIPA Ontology Service Recommendations (<http://www.fipa.org/>). It uses classes to describe concepts and predicates and fixes them as part of the application ontology. In this way, an ontology is quickly generated with an immediate underlying implementation.

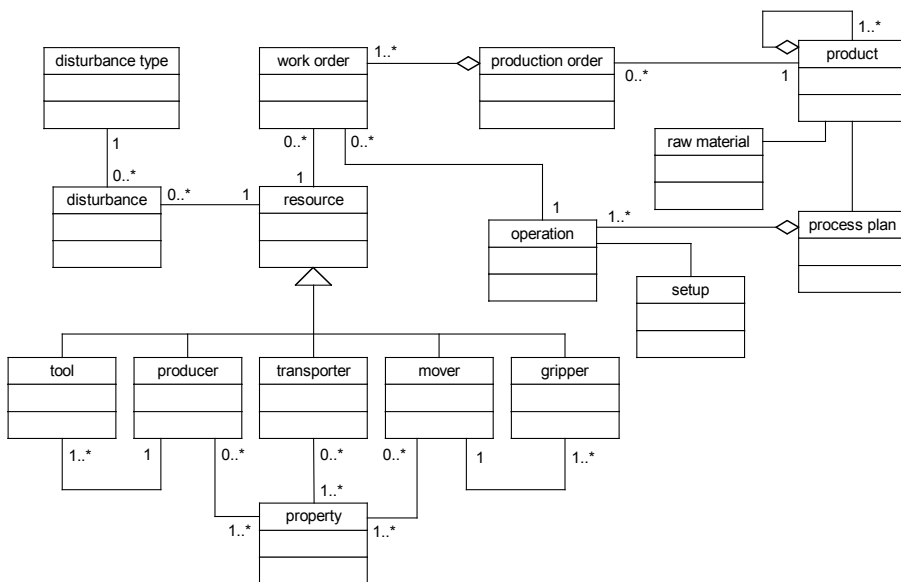


Figure 1. Manufacturing Ontology in the ADACOR Architecture

The manufacturing ontology used in ADACOR is developed through the definition of a taxonomy of manufacturing components, which contributes to

the analysis and formalization of the manufacturing problem (these components are mapped into a set of objects, illustrated in the UML class diagram of

#. Foundations for a core Ontology of Manufacturing 39

). For this, one must fix the vocabulary used by the distributed entities over the ADACOR platform, isolate the ADACOR-concepts, the ADACOR-predicates and -relations, the ADACOR-attributes of the classes, and the meaning of each term. Note that not all ADACOR concepts find a place in Figure 1. The diagram is restricted to the relationships between simple manufacturing components used by the manufacturing control system. For example, a production order may index one customer order or an aggregation of these although this relationship and the latter concept are not shown.

4.1 ADACOR Concepts

ADACOR-concepts are expressions that hold for complex entities whose structure can be defined in terms of classes or objects. The main concepts in the ADACOR architecture, Figure 1, are informally described as follows:

- Product: entity produced by the enterprise (it includes sub-products).
- Raw-material: entity acquired outside the enterprise and used during the production process, e.g. blocks of steel, nuts and bolts (unless produced internally).
- Customer order: entity received by the enterprise from a customer requesting products.

- Production order: entity obtained by aggregating customer and forecast orders.
- Process Plan: description of a sequence of operations (for producing a product) with temporal constraints like precedence of execution.
- Operation: a job executed by one resource like drilling, maintenance, and reconfiguration of resources.
- Work order: entity that describes the production of a product by listing the operations and processing time, participants (e.g. type and number of resources), priority, scheduled dates, state and quantity.
- Resource: entity that can execute a certain range of operations as long as its capacity is not exceeded. Producer, mover, transporter, tool, and gripper are specializations of resource and inherit its characteristics³.
- Disturbance: unexpected event, like machine failure or delay, that degrades the execution of a production plan.
- Setup: set of actions that it is necessary to execute in order to prepare a manufacturing resource for the execution of a range of operations.
- Property: an attribute that characterizes a resource or that a resource should satisfy to execute an operation.

In agent-based or holonic manufacturing control approaches, the control is achieved by the interaction between distributed entities, i.e. the agents or the holons. An agent or holon in ADACOR is an entity that represents manufacturing components like resources, products or orders.

³ Here human operators are not considered among the resources of the system.

4.2 ADACOR Predicates

Predicates establish relationships among concepts, for instance:

- $\text{ComponentOf}(x,y)$: product x is a component of product y .
- $\text{Allocated}(x,y,t)$: operation x is allocated to resource y at time t .
- $\text{Available}(x,y,t)$: resource x is available at time t for operation y .
- $\text{RequiresTool}(x,y)$: execution of operation x requires tool y .
- $\text{HasTool}(x,y,t)$: resource x has tool y available in its magazine at t .
- $\text{HasSkill}(x,y)$: resource x has property (skill) y .
- $\text{HasFailure}(x,y,t)$: a disturbance x occurred in resource y at time t .
- $\text{Proposal}(x,y,w,z,u)$: the entity x proposes to the entity y the execution of the work order w with location u and charging the price z .
- $\text{Precedence}(x,y)$: operation x requires previous execution of y .
- $\text{UsesRawMaterial}(x,y)$: production order x uses raw material y .
- $\text{RequestSetup}(x,y)$: operation x needs the execution of setup y .
- $\text{HasProcessPlan}(x,y)$: production of x requires process plan y .
- $\text{OrderExecution}(u,x,w,y)$: operation u is listed in process plan w (describing production of y) for production order x .
- $\text{HasRequirement}(x,y)$: operation x requires property y .
- $\text{HasGripper}(x,y,t)$: resource x has gripper y in its magazine at time t .
- $\text{ExecutesOperation}(x,y)$: work order x includes operation y .

If these predicates are available, an *agenda* can be defined as a set of $\text{Allocated}(x,y,t)$ and $\text{Available}(x,y,t)$ predicates.

4.3 Attributes of ADACOR Concepts

Attributes are values relative to properties of concepts. Here is a list of properties (in brackets an example of measure units) associated with the skills of a resource or the requirements of an operation:

- Axes: a non-negative integer, e.g. the number of axes of a machine.
- ProcessingType: a type of processing e.g. turning, milling, or drilling.
- Repeatability: a non-negative integer, it gives an indication about the precision of the machine (expressed in mm).
- FeedRate: a positive rational number, it gives the feed rate of a specific axis (expressed in mm/rot).
- SpindleSpeed: a range of non-negative integers, it gives the spindle speed in the form [min, max] (expressed in rpm).
- CuttingSpeed: a positive rational number, it gives the cutting speed (expressed in mm/s).
- Tailstock: a range of non-negative integers, it gives the size in form [min,max] of pieces that the machine can process (expressed in mm).
- Payload: positive integer, it gives the maximum load of the robot that guarantees the repeatability (expressed in kg).
- MaxReachability: positive integer, it gives the work volume of the robot (expressed in mm).
- Autonomy: non-negative integer, it gives the amount of time that an autonomous vehicle can work without the need to re-fill its batteries (expressed in hours).

- MagazineCapacity: non-negative integer, it gives the number of tools or grippers that the magazine of a machine or robot can store.

5. THE DOLCE FORMAL ONTOLOGY

In section 3.3, a number of foundational ontologies have been introduced. This section focuses on the DOLCE ontology and presents those features that are most relevant for a manufacturing core ontology. The interested reader can find in (Masolo et al., 2003) the motivations for this ontology and a throughout discussion of technical aspects.

5.1 DOLCE from the Manufacturing Perspective

The Descriptive Ontology for Linguistic and Cognitive Engineering, DOLCE, concentrates on *particulars*, that is, roughly speaking, objects (physical or abstract), events, and qualities. The ontology does not attempt to provide a taxonomy of properties and relations and these are included in the system only if crucial in characterizing particulars.

The DOLCE ontology provides a good framework for the manufacturing area: it adopts the distinction between objects like products and events like operations; it includes a useful differentiation among individual qualities, quality types, quality spaces, and quality values; it allows for fine descriptions of properties and capacities; and it relies on a very expressive language, namely first-order modal logic. Because of these features, the formalization of categories like physical object, agent, and process can be

done following the corresponding notions as used in the manufacturing domain. Furthermore, in DOLCE the user can choose and characterize the qualities needed in the application which provides a great level of freedom and facilitates update and maintenance. From the implementation viewpoint, lightweight versions of DOLCE are available in LOOM, DAML+OIL, RDFS, and OWL and the full system is implemented in CASL (see <http://www.brics.dk/Projects/CoFI/CASL.html>) with connections to theorem provers and graphical tools.

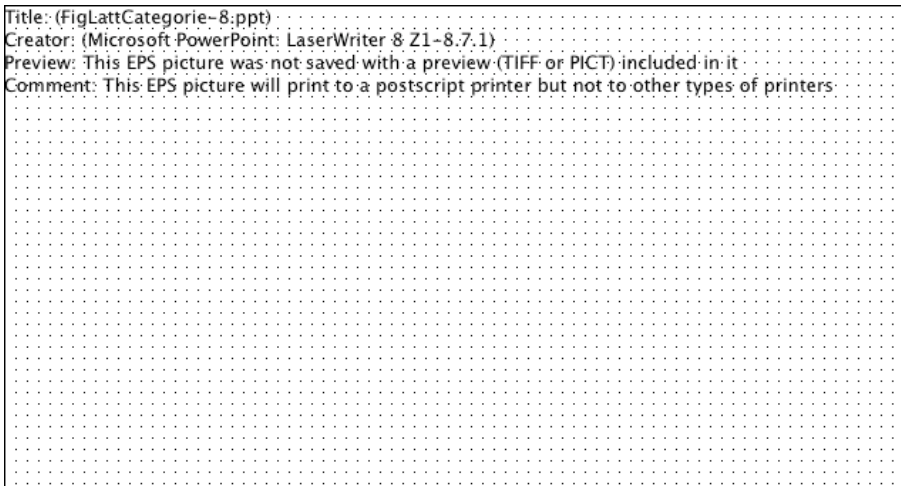


Figure 2. Taxonomy of DOLCE basic categories (Masolo et al., 2003).

The development of DOLCE has been explicitly influenced by natural language and cognitive considerations. This explains in part the adoption of a *multiplicative approach*, that is, the assumption that different entities can be co-localized in the same space-time. For example, a drilling machine and

the amount of matter that forms it are captured in DOLCE as two distinct entities (as opposed to different aspects of the same entity). The reason lies on the different set of properties that these entities enjoy: the drilling machine ceases to exist if a radical change of shape occurs (e.g., when it is crashed and it cannot be repaired) while its amount of matter is not affected.

The ontology uses *endurant* for objects like “gripper” or “plastic”, and *perdurant* for events like “making a hole”, “moving a steel block” and the like. The term ‘object’ is used in the ontology to capture a notion of unity as suggested by the partition of the class “physical endurant” into classes “amount of matter”, “feature”, and “physical objects” (Figure 2). Both endurants and perdurants are associated with a bunch of *qualities*, the list of qualities may depend on the entity: shape and weight are usually taken as qualities of endurants, duration and direction as qualities of perdurants. Roughly speaking, in DOLCE an *individual quality* is a quality associated with *one and only one* entity; it can be understood as the particular way in which that entity instantiates the corresponding property. For example, in DOLCE the endurant *Gripper_321* (a physical machine) has its own individual instantiation of property “having weight”. This instantiation is the *individual weight-quality* of *Gripper_321*. *Gripper_321* may have several individual qualities, each related to a different property: the individual weight-quality, the individual shape-quality, the individual color-quality, etc. Recall that DOLCE gives freedom in choosing the individual qualities that are associated with an entity. This is important since particular properties like “having light sensors” are relevant in specific applications

only. Yet, they should not be ruled out from the start nor forced to be always considered. The change of an endurant in time is explained through the change of some of its individual qualities. For example, with the substitution of a component, *Gripper_321* may increase its weight from a to b , then the individual weight-quality of *Gripper_321* changes since that individual quality was initially associated with a and then with b . Note that a and b should not be considered weight measures like, say, 5 kg. They are elements of a space called *quality space* or, in this specific example, the *weight-quality space*. We will discuss quality spaces below. First, note that the substituted component of *Gripper_321* must *not* be essential to the gripper. The substitution of an essential part would destroy *Gripper_321* and generate a new one. Finally, note that the gripper cannot exist without its individual qualities: DOLCE forces a strict existential dependence between individual qualities and their hosts.

The example of the gripper makes clear that the “position in the quality space” of an individual quality can change over time. DOLCE calls such positions *qualia* (*quale* in singular form). A *quality space* for a property is the collection of all possible qualia (positions) that an individual quality can assume. Suppose that in an application an endurant is either heavier, equal, or lighter than another endurant α . Then, an admissible weight-quality space for that application has *at least* three distinguished positions: one is the position taken by the individual weight-qualities of endurants lighter than α , a second position is taken by the individual weight-qualities not distinguishable from the individual weight-quality of α , and the third is the

position of the individual weight-qualities of the heavier endurants. If other considerations seem to require a different system, one can assume that the quality space (the set of positions) is more complex, for instance the set of non-negative real numbers. This latter case can be described in DOLCE by positioning the individual weight-quality of α at some positive real x and the individual quality of an endurant lighter than α to the sub-region $\{z \in \mathfrak{R}^+ \mid z < x\}$. (Here the position is the whole region, not just a point in it. Mereological principles are used to map between regions and points when needed.) Similarly, the individual quality of an endurant heavier than α is positioned at the sub-region $\{z \in \mathfrak{R}^+ \mid z > x\}$. The mapping between the two weight-quality spaces just described is trivial. Note that neither measurement methods nor units have been used, these must be introduced explicitly.

Finally, DOLCE is actively compared to other foundational ontologies at different levels of formality (Masolo et al., 2003; Martin, 2003). Since DOLCE is included in merging initiatives, the core ontology here proposed is likely to be automatically connected to any other manufacturing ontology developed for interoperability.

5.2 Categories and Relations in DOLCE

The categories of Figure 2 relevant to our work are here introduced.

- $ED(x)$, $PED(x)$ stand for “ x is an endurant” and “ x is a physical endurant”, respectively, with the latter is a subclass of the first.

- $NAPO(x)$ stands for “ x is a non-agentive physical object”, i.e., endurants that have spatial and temporal location but not intentions, believes, or desires, like “products” and “production orders”.

In the manufacturing domain, one needs to deal with a variety of operations (jobs). First, note that here an operation is a precise event or happening, that is a precise perdurant, and not a type of perdurants. (Types are introduced through the use of descriptions. This crucial distinction will be developed at another stage of this work. See also section 6). An operation in DOLCE is said to be *homeomeric* if every temporal part of it is itself an operation of the same “type”. For instance, a “milling” operation during interval t is homeomeric since if one divides this interval in two parts, say, t_1 and t_2 , the sub-operation during t_1 is still a milling operation and so is the sub-operation during t_2 . This does not hold for “setup” operations. A setup operation requires the completion of a process which is obtained once a specific state is reached. If this does not happen, the setup does not occur. Thus, if a setup operation is divided in two temporal parts as before, only one of the two sub-operations (if any) can be considered a setup operation. This and similar distinctions drive the ontological classification of the ADACOR notions and are captured by the DOLCE predicates below.

- $PD(x)$ stands for “ x is a perdurant”.
- $ACH(x)$ stands for “ x is an achievement”, i.e., perdurants that are *anti-cumulative* (summing two achievements one does not obtain an

achievement) and *atomic* (they do not have temporal parts). E.g., the “completion of a reconfiguration”⁴.

- $ACC(x)$ stands for “ x is an accomplishment”. These are *non-atomic* perdurants since they have temporal parts. E.g. “machine reconfiguration”. The fact that the sum of two “machine reconfigurations” is not a “machine reconfiguration” itself shows that accomplishments are *anti-cumulative*.
- $ST(x)$ stands for “ x is a state”, i.e., *cumulative* perdurants like “drilling” (the sum of two drilling operations is again a drilling operation). These perdurants are also *homeomeric*.
- $qt(q,x)$ stands for “ q is an individual quality of x ”.
- $ql(r, q), ql(r,q,t)$ stand for “ r is the quale of the perdurant’s quality q ”, “ r is the quale of the endurant’s quality q during time t ”, respectively.

6. THE ALIGNMENT ADACOR - DOLCE

DOLCE provides a distinct category for each type of entity in ADACOR.

Beside the distinction between endurants and perdurants, descriptions are modeled explicitly as abstract entities, and properties are rendered through the associated qualities. In this section the ADACOR concepts, predicates and attributes are checked from an ontological perspective and classified in DOLCE. For lack of space, not all the notions of section 4 are included. Nonetheless, the overall framework should be clear from the cases below.

⁴ Note the distinction between “completion of a reconfiguration” and “reconfiguration”. Only

6.1 ADACOR - DOLCE: Endurants

A crucial point is the distinction between endurants and their descriptions. An example is given by the concept of “order”, let it be “customer order”, “production order” or “work order”. In the manufacturing enterprise “order” is at the same time the physical support for some data (a physical object like a sheet of paper or a part of a computer device) and the description of an entity or event (the description of an operation that must be executed or of a product that must be produced). Since “order as a physical object” and “order as a description” have different properties (if one can take a physical object from one office to another, it makes no sense to take an abstract entity from an office to another; likewise a product can conform to a description but not to a paper)⁵, it is necessary to make sure that the formalization keeps them distinct.

For each ambiguous ADACOR concept, two predicates are introduced in DOLCE; one referring to endurants (in this case the very same expression is used), the other referring to descriptions (in this case the superscript ‘*D*’ is added). Here it is assumed that a description is considered as long as recorded in some physical object, e.g. in a document about product specifications. (The study of descriptions is not presented here except for one minor case.)

the first is an achievement.

⁵ Here one should refrain from exploiting the ambiguities of natural language (one of the reasons for employing formal ontology). In a given context one finds meaning even for sentences like “take this description to the management office” or “this bolt conforms to the paper they gave me”. If sentences are interpreted contextually, the communication cannot be reliable unless there is only one context.

Products, resources and orders (as physical durants) are non-agentive entities, thus they are naturally classified as *NAPO*

$$(Product(x) \vee Resource(x) \vee Order(x)) \rightarrow NAPO(x)$$

At the level of descriptions,

$$(Product^D(x) \vee Resource^D(x) \vee Order^D(x)) \rightarrow AB(x)$$

where *AB* is the predicate that characterizes abstract entities in DOLCE (entities neither in space nor in time).

Since raw-material may refer to physical objects (bolts, lenses, etc.) as well as to amounts of matter (water, sand or gasoline), this concept is mapped to the category *PED* which includes both

$$Raw_material(x) \rightarrow PED(x)$$

The constraint $Raw_material(x) \rightarrow (POB(x) \vee M(x))$ would be too restrictive since it requires any raw-material to be either a physical object or an amount of matter (the two class are disjoint in DOLCE). That would exclude raw-material composed of physical objects and some amount of matter. Instead, it is necessary to add the restriction that features are not raw-material

$$Raw_material(x) \rightarrow \neg F(x)$$

Note that the notion of raw material is not ontological. A company that produces clothes and that buys buttons from another producer classifies the buttons as raw-material. Indeed, buttons are here parts of the produced items without being products themselves. However, the very same items are products for the button producer. This discrepancy is only apparent since

ontologically the items (call them raw-material or products) have the same individual qualities in all contexts.

Some ADACOR concepts, like “Order” or “Resource”, are totally determined in terms of more specialized entities also in ADACOR. In these cases, the formalization lists which entities these concepts subsume. In particular, the “Order” and “Resource” are partitioned as follows

$$Order(x) \leftrightarrow (Production_order(x) \vee Customer_order(x) \vee Work_order(x))$$

$$Resource(x) \leftrightarrow (Producer(x) \vee Mover(x) \vee Transporter(x) \vee Tool(x) \vee Gripper(x))$$

6.2 ADACOR - DOLCE: Perdurants

Most of the entities in ADACOR are perdurants (or descriptions of perdurants) since they identify activities or states. If the classification of “Operation” as generic perdurant is immediate, the notion of “Disturbance” is more involved and will be discussed below together with “Delay” and “Failure”. As for "Completion", it marks the end of an event and thus it is an achievement. The remaining operations are divided in two groups: stative perdurants and accomplishment perdurants.

$$Operation(x) \rightarrow PD(x)$$

$$Disturbance(x) \rightarrow ACH(x)$$

$$(Transportation(x) \vee Turning(x) \vee Drilling(x) \vee Milling(x)) \rightarrow ST(x)$$

$$(Setup(x) \vee Reconfiguration(x) \vee Inspection(x) \vee Maintenance(x) \vee Assembly(x) \vee Production(x)) \rightarrow ACC(x)$$

It is natural to consider “transportation” as a state: since all the temporal parts of a transportation event can be classified as transportations themselves, this type of event falls in the class of stative perdurants (*ST*). A similar argument holds for “turning”, “drilling”, and “milling” since these are relatively simple perdurants. More specialized operations, for instance operations where it is necessary to distinguish explicitly different phases of execution (say, to resume properly after a failure event), may require the notion of process. As of now, this kind of operations is not present in ADACOR. The remaining operations are all implicitly characterized by a notion of “final state” and, consequently, classified as accomplishments.

“Operation”, “Disturbance”, and “Reconfiguration” are characterized by more specialized notions as follows

$$Disturbance(x) \leftrightarrow (Failure(x) \vee Delay(x))$$

$$Operation(x) \leftrightarrow (Completion_of_setup(x) \vee Reconfiguration(x) \vee Inspection(x) \vee Setup(x) \vee Maintenance(x) \vee Turning(x) \vee Production(x) \vee Milling(x) \vee Transportation(x) \vee Assembly(x) \vee Drilling(x))$$

$$Reconfiguration(x) \leftrightarrow (Addition_of_new_resource(x) \vee Change_of_layout(x) \vee Removal_of_resource(x) \vee Change_of_resource_capability(x))$$

Correctly, ADACOR considers “Operation” and “Disturbance” as disjoint notions, that is, no entity is both an operation and a disturbance.

$$Operation(x) \rightarrow \neg Disturbance(x)$$

There is a misalignment between the notion of “setup” as an operation (above) and the concept of “setup” shown in Figure 1. In some cases, “setup” is seen as a requirement for other operations and this justifies its addition as a separate entry in Figure 1. The status of “being a requirement” can be captured ontologically through a standard *precedence* relation.

“Delay”, “Disturbance”, and “Failure” are special kind of events in ADACOR. A disturbance is an unexpected event: machine failure or machine delay are the only examples of disturbances considered. These events affect the scheduled production plan. When an operation is being executed, several different scenarios can be expected: (1) the resource finishes the execution of the operation within the estimated time interval, (2) the resource fails and it cannot finish the operation (a failure has occurred) or (3) the operation is delayed (a delay has occurred). Thus, failures and delays are perdurants and machines participate in them. Clearly, a failure is a kind of achievement (the event at which a production plan rescheduling is requested). The classification of “Delay” is similar although it might be less obvious. First it is important to understand that not all holdups are delays. For a delay to occur, it is not enough to have an operation postponed or retarded. What matters is the satisfaction of the temporal constraints set by the production plan. A delay occurs only when it is acknowledge that the production plan cannot be satisfied. Thus, it marks a state where a production plan rescheduling is requested and so it is an achievement. In short, the distinction between “Failure” and “Delay” is based on the cause

that brought to the rescheduling request, not on the type of perdurant these notions refer to. For the sake of completeness, regarding the anti-cumulativeness property note that the sum of two delays is not itself a delay since it does not correspond to a “single” rescheduling request.

6.3 ADACOR - DOLCE: Qualities

In the terminology of DOLCE, skills are qualities of endurants. For each type of skill it is introduced a quality space and, for each endurant that has that skill, an individual quality specific to that entity. The quale gives a classification of that endurant with respect to the given skill. Since skills require the introduction of all these different elements, details about the ontological analysis of skills will be presented in a dedicated paper. Here it suffices to give a guiding example through the attribute “Autonomy”.

First, note that skills are not necessarily *ontological* qualities. The notion of “Autonomy” is important in the manufacturing area and it is considered as an independent property in ADACOR. Thus, it is included in the proposed manufacturing ontology. However, in other applications it might be given as a derived property (depending on, say, batteries power and energy consumption). DOLCE can deal with both cases and it furnishes the tools to coherently relate the different characterizations.

The autonomy of a resource measures how long it can work without the need to re-fill its batteries. If *AutL* is the class of individual autonomy-qualities, then the following constraint says that “Autonomy” is a quality defined for resources only

$$AutL(q) \rightarrow \exists x (qt(q,x) \wedge Resource(x))$$

Literally the formula states that each individual autonomy-quality is a quality of a resource. The uniqueness of the resource is derived from the formalization of DOLCE itself.

The specific relations “ q is the autonomy-quality of resource x ” and “resource x has autonomy-quality d at time t ” are not part of the language and can be defined as follows

$$Autonomy(q,x) =_{def} Resource(x) \wedge AutL(q) \wedge qt(q,x)$$

$$Autonomy(d,x,t) =_{def} Resource(x) \wedge \exists q (Autonomy(q,x) \wedge ql(d,q,t))$$

Assume now that f is a function from the autonomy-quality space to the non-negative integers obtained by fixing some standard measurement method and unit for this property. Also, assume autonomy is expressed in hours and that relation $Executes(x,y,t)$, which reads “resource x starts executing operation y at time t ”, is given (see for instance (Borgo and Leitão, 2004)). Then, the language allows us to put constraints on the autonomy capacity of a resource by

$$Operation_requires_autonomy(x,y) =_{def}$$

$$Operation(x) \wedge \forall z,t, d (Executes(z,x,t) \wedge Autonomy(d,z,t) \rightarrow f(d) \geq y)$$

$$Operation_requires_autonomy(milling_\alpha, 3)$$

From the definition x is an operation that requires an autonomy of at least y to be executed, and the other formula constrains operation $milling_\alpha$ to be executed by resources with at least 3 hours autonomy.

6.4 The ADACOR - DOLCE Notion of Component

This part focuses on the ADACOR concept “ComponentOf” (see section 4.2) and the notion of process plan.

From section 2.1, the structure of a product is included in the product model. Assume that the production of a product consists simply in assembling its components. A component may be complex, i.e., itself decomposable into simpler components, or atomic. The idea is that all the elements that are assembled at some point of the production process are components of the product itself. The ADACOR notion “ComponentOf” is needed to provide this composition-hierarchy in the product model. The goal is to capture this informal description from an ontological viewpoint in order to avoid misinterpretations of the product model. For this, the predicate *Component_of* is introduced in DOLCE

$$Component_of(x,y) \rightarrow ((Product(x) \vee raw_material(x)) \wedge Product(y))$$

$$Component_of(x,y) \rightarrow \neg Component_of(y,x) \quad (anti-symmetric)$$

$$Component_of(x,y) \wedge Component_of(y,z) \rightarrow \\ \rightarrow Component_of(x,z) \quad (transitive)$$

$$\forall x \exists y (Product(x) \wedge \neg raw_material(x)) \rightarrow Component_of(y,x) \\ (if\ x\ is\ a\ produced\ product,\ then\ it\ has\ components)$$

The first condition says that if x composes y , then x is either a product or raw material while y is a product. The second condition implies that a component cannot be a component of itself. The next formula states

transitivity: a component of a component of z is also a component of z . Finally, a constraint is added to the effect that only products which are also raw material have no component. The inverse relation “ x has component y ”, call it *Has_component*, is given by

$$\text{Has_component}(x,y) \leftrightarrow \text{Component_of}(y,x)$$

Regarding the notion of process plan, a graphical and mathematical representation of process planning information can be done using standard graph theory as in (Cho and Wysk, 1995). This allows us to represent processing precedence, alternative sequences and parallel actions. (Cho and Wysk, 1995) introduces an AND/OR based graph to represent the operations and their precedence relationship (Figure 3).

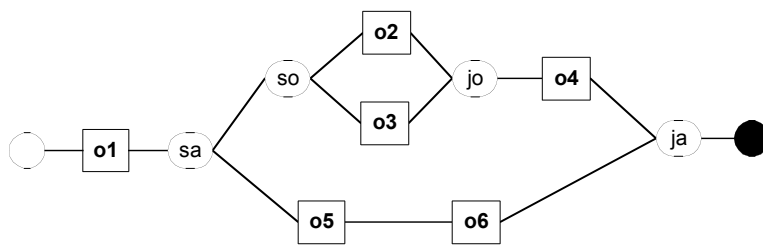


Figure 3. A Process plan representation example.

There are five types of nodes: *operation*, *split-or*, *split-and*, *joint-or*, and *joint-and*. All paths following a *split-and* type node must be processed since they are always necessary for the production. A *joint-and* type node brings multiple paths back together after a *split-and* type node. Only one path following a *split-or* type node must be selected for execution. In this way,

one can represent operations alternatives. A *joint-or* type node is required to bring multiple paths together after a *split-or* type node. Figure 3 exemplifies an AND/OR based graph, where the process plan comprises the execution of operation **o1** and the execution of one of two alternative set of operations: the first one comprises the execution of operation **o2** or **o3** followed by the execution of **o4**, and the second one comprises the execution of operations **o5** and **o6**.

Below a representation in DOLCE of the temporal sequence of operations in Figure 3 is given. To keep the formula simple, the “disjoint or” connective (indicated by symbol $\dot{\vee}$) is used: a formula of form $\alpha \dot{\vee} \beta$ reads “either α or β is true but not both”. Among the nodes, only operations need to occur in the formalization. Terms o_1, \dots, o_6 refer to **o1**, ..., **o6**, respectively as occurring in Figure 3. Recall that the relation $Executes(x,y,t)$ stands for “resource x executes operation y at time t ”. In the formula, x_i stands for a resource (say, a machine) and t_i for the initial time of the execution⁶.

$$\begin{aligned}
& \exists x_1, \dots, x_6, t_1, \dots, t_6 \\
& (t_1 < t_2, t_3, t_5 \wedge t_2, t_3 < t_4 \wedge t_5 < t_6 \wedge t_2 = t_3) \quad (\text{temporal constraints}) \\
& \wedge [Executes(x_1, o_1, t_1) \quad (\text{starting condition}) \\
& \quad \wedge (Executes(x_2, o_2, t_2) \dot{\vee} Executes(x_3, o_3, t_3)) \quad (\text{do either } o_2 \text{ or } o_3) \\
& \quad \wedge Executes(x_4, o_4, t_4) \wedge Executes(x_5, o_5, t_5) \wedge Executes(x_6, o_6, t_6)]
\end{aligned}$$

⁶ One can restate the formula using time intervals or adding more time constraints by taking into account the duration of the different operations.

7. CONCLUSIONS AND FUTURE WORK

The ADACOR manufacturing ontology, described in the section 4, was implemented as part of a multi-agent manufacturing control system by using the JADE (Java Agent Development Framework) framework. The experience gained during the development phase, highlighted the difficulties to build, maintain and modify proprietary ontologies to be used by heterogeneous manufacturing control applications, especially those built upon distributed approaches such as multi-agent systems. This problem pushed for new approaches in the development of manufacturing ontologies to simplify the effort to build, maintain and modify the ontologies. The adoption of an established foundational ontology was suggested to overcome this problem and to improve the consistency of the overall system.

In this paper, a classification of ADACOR concepts according to the DOLCE foundational ontology has been proposed resulting in the core ontology of section 6. This ontology improves and extends (Borgo and Leitão, 2004) and can be used within actual implementations of ADACOR and adapted to other architectures. The formal expressions generated by this ontology can be furnished together with the data exchanged among the agents and holons in this way guaranteeing the correct interpretation of the data. The formal expressions can be obtained either at development time (for general data) or at run-time throughout the available lightweight versions of DOLCE. A quick check through the same lightweight version of the system ensures the correct meaning of the data is understood by the receiver.

Roughly, every time a term or a set of data are ontologically ambiguous, a flag is set to mark the data; both the sender and receiver can check the possible meanings and start a negotiation process if needed.

When the ontology is completed to cover all the concepts and relations of ADACOR, a series of tests will be executed to evaluate in real applications the reliability of the core ontology as well as its usefulness in the manufacturing area. In spite of some efforts to develop ontologies in areas related to manufacturing, as of today no available (or even proposed) formal ontology seems capable to cover the all domain.

The core ontology presented in this paper is well-founded because built according to a foundational ontology (DOLCE), because adopting formal semantics, and because following the methodology of formal ontology. This fact makes the proposed core ontology conceptually transparent and semantically explicit, two conditions crucial for information communication, sharing, and retrieval. However, this system is only an initial step in the realization of our goal since only entities performing the manufacturing scheduling and control operations have been considered and no test in real applications has been carried out yet. Also, specifications of additional information beyond process data, for instance resource commitments, costs, delivery times and machine failures need to be investigated further.

Manufacturing enterprises normally suffer for lack of generality, reliable intra/inter communications and re-usability of their systems. Since the proposed here approach presents some innovations, such as formal specification, independence from implementation and platforms, generality

and re-usability, if successful it will support the development of heterogeneous and distributed manufacturing scheduling and control systems, allowing a later integration of information systems at intra-enterprise and inter-enterprise levels.

ACKNOWLEDGEMENTS

The Provincia Autonoma di Trento (IT) funded one author through the projects “Logical Instruments for Ontology Analysis” and “Mostro”.

8. REFERENCES

- Bertolazzi, P. Krusich, C. and Missikoff, M., 2001, An Approach to the Definition of a Core Enterprise Ontology: CEO, OES-SEO 2001, International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations, Rome, September.
- Borgo, S. and Leitão, P., 2004, The role of Foundational Ontologies in Manufacturing Domain Applications, in R. Meersman and Z. Tari (eds.), OTM Confederated International Conferences, ODBASE 2004, LNCS 3290, Springer, pp. 670–688.
- Brunnermeier, S. B. and Martin, S. A., 1999, Interoperability Cost Analysis of the U.S. Automotive Supply Chain (Planning Report #99-1), Technical report, NIST, Research Triangle Institute (also available at <http://www.nist.gov/director/progofc/report99-1.pdf>).
- Catron, B. and Ray, S., 1991, ALPS: A Language for Process Specification, *International Journal of Computer Integrated Manufacturing*, **4**(2): 105–113.
- Cho, H. and Wysk, R., 1995, Intelligent Workstation Controller for Computer Integrated Manufacturing: Problems and Models, *Journal of Manufacturing Systems*, **14**(4):252–263.
- European Commission, Manufuture, 2004, A Vision for 2020, Assuring the Future of Manufacturing in Europe, Report of the High-level Group.

- Fadel, F., Fox, M. and Gruninger, M., 1994, A Generic Enterprise Resource Ontology, In 3rd IEEE Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises.
- Fellbaum, C. (ed.), 1998, *WordNet An Electronic Lexical Database*, Bradford Book.
- Guarino, N., 1998, Formal ontology in information systems, In N. Guarino (ed.), Proceedings of the 1st Intern. Conf. on Formal Ontology in Information Systems, IOS Press, pp. 3-15.
- Heller, B. and Herre, H., 2003, Ontological Categories in GOL, *Axiomathes*, (14)1:57-76.
- Koestler, A., 1969, *The Ghost in the Machine*, Arkana Books, London.
- Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A. and Yost, G., 1998, The PIF Process Interchange Format and Framework, *Knowledge Engineering Review*, 13(1):91-120.
- Leitão, P., Colombo, A. and Restivo, F., 2005, ADACOR, A Collaborative Production Automation and Control Architecture, *IEEE Intelligent Systems*, 20(1):58-66.
- Martin, P., 2003, Correction and Extension of WordNet 1.7, Proceedings of the 11th Intern. Conf. on Conceptual Structures, LNAI 2746, Springer, pp. 160-173.
- Masolo, C., Borgo, S., Gangemi, A., Guarino, N. and Oltramari, A., 2003, Ontology Library (Wonder-Web Deliverable D18), Technical report, Laboratory for Applied Ontology, ISTC-CNR, (also at <http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf>).
- Oltramari, A., Gangemi, A., Guarino, N., and Masolo, C., Restructuring WordNet's Top-Level: The OntoClean approach, Workshop Proceedings of OntoLex 02, Ontologies and Lexical Knowledge Bases, LREC2002, May 27, 2002, pp. 17-26.
- Schlenoff, C., Knutilla, A. and Ray, S., 1996, Unified Process Specification Language: Requirements for Modeling Process, In NIST, Interagency Report 5910, Gaithersburg MD.
- Uschold, M., King, M., Moralee, S. and Zorgios, Y., 1998, The Enterprise Ontology, *Knowledge Engineering Review, Special Issue on Putting Ontologies to Use*, 13(1):31-89.
- van der Vet, P., Speel, P.-H. and Mars, N., 1994, The PLINIUS Ontology of Ceramic Materials, In ECAI'94, Workshop on Comparison of Implemented Ontologies.