The Role of Foundational Ontologies in Manufacturing Domain Applications

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.[1] Although ontology has gained wide attention in the area of information systems, a criticism typical of the early days is still rehearsed here and there. Roughly, this criticism says: general ontologies are not suited for real applications. We believe this is the result of a misunderstanding of the role of general ontologies since, we claim, even foundational ontologies (the most general and formal ontologies) have a crucial role in building reusable, adaptable and transparent application systems. We support this view by showing how foundational ontologies can be used in the manufacturing control area. Our approach (partially presented here through an example) provides a domain-specific ontology which is explicitly designed for applications, theoretically organized by a foundational ontology, driven by the application field for all intents and purposes, suitable for communication across different applications.

1. Introduction

In information science, ontology stands for a knowledge engineering artifact constituted by an interpreted language plus a set of explicit assumptions; its goal is to describe a certain reality of interest [1]. Taking the degree of semantic precision as basic metric, ontologies form a spectrum with simple glossaries and thesauri on one side and rich logical theories on the other. Ontologies resembling glossaries and thesauri, like WordNet [2], are helpful in organizing databases and protocols where only terminological services are needed. When sophisticated knowledge structures become necessary, much richer systems should be applied, e.g. those described in the Library of Foundational Ontologies [3]. Since these rich ontologies do not enjoy nice computational properties, to maintain effective computability one separates representation and reasoning issues by adopting two types of ontology: a foundational ontology that provides the full system and is applied at development-time, and a lightweight ontol-

[1] The first author has been partially supported by the Provincia Autonoma di Trento. The authors would like to thank Claudio Masolo and the anonymous referees for their comments.

ogy (a simplified version of the previous) that furnishes an efficient, although minimal, system used at run-time. Adopting this distinction, we concentrate on foundational ontologies and show their role in generating reliable representation systems.

Ontologies are nowadays quite common in information systems[2], however in the literature one hardly finds real applications developed on top of foundational ontologies. There are several reasons for this; foundational ontologies are relatively new and only in the last few years well axiomatised and justified systems have been proposed. Moreover, the development of application systems based on these ontologies is demanding so that few projects have undertaken this challenge [4]. More often, researchers focus on goals that seem to be just pieces of the process we envision [5]. Our hope is that a consistent deployment of foundational ontologies in a traditional and well established area like the manufacturing domain will foster a better understanding of these theoretical tools and of the advantages of systems based on them.

The majority of the ontologies so far developed in Artificial Intelligence express simple relationships among terms (primarily taxonomies) perhaps with some set of formal constraints (formal ontologies). Foundational ontologies stand out as specialized logical theories (a subclass of formal ontologies) not limited to particular domains and developed with the intention of characterizing explicitly a viewpoint on the "reality": the aim is to *capture formally the (intended) meaning of the adopted language*. Among the advantages in applying these, they drastically reduce misinterpretation of the knowledge base (semantic explicitness) and make information sharing reliable even in communication among untrained users and software agents (conceptual transparency). However, these ontologies are trustworthy only if based on a careful and detailed ontological analysis of (a viewpoint of) reality, a lengthy and time-consuming process which must be coupled with a rigorous logical characterization. Furthermore, they must guarantee the coverage of general and disparate concepts, allow for subtle distinctions, and make space for the specific interests of potential users. Indeed, the primitive notions of the ontology and the constraints stated to characterize them form a richly structured framework where entities, concepts, and relations of the domain at stake must find a place. In other terms, in deploying a foundational ontology one assumes that this system covers (perhaps only implicitly) all possible concepts and relations of interest. Furthermore, one accepts the view that any element in the domain can be captured in logical terms within this framework and that any expression means whatever the formal semantics states. Some researchers maintain that these assumptions are too strong and that no ontology can deliver such a characterization of the language. Consequently, they prefer to use weak terminological ontologies claiming that foundational ontologies are too brittle theoretical tools and, as such, not suited for application domains [6].

We disagree with this general standpoint. We believe this criticism is the result of a misunderstanding of the significance of foundational ontologies in application domains. It is widely recognized that foundational ontologies furnish an important tool for establishing links and comparisons among domain ontologies, especially when the focus is on communication and standardization. Indeed, they make explicit the philosophical, cognitive, and linguistic commitments the different systems have. However, this is not the only role such ontologies can play. On the contrary, we claim that

[2] http://www.semanticweb.org/

foundational ontologies are crucial for the very development of domain-specific ontologies and, as such, they are profitable in applications. The case we present corroborates this view by applying foundational ontologies in modelling problems, by showing the role of foundational ontologies in building applicative systems, and by highlighting their relevance. For this, we chose to work in a domain (manufacturing enterprise) that has proven to be quite successful in modelling production processes but that shows some weakness in the area of information integration and management.

Organization of the paper. Section 2 gives an overview of the manufacturing domain and section 3 concentrates on the ADACOR architecture with its terminological system. In section 4, we discuss interoperability issues and briefly look at different foundational ontologies available in the literature motivating our choice to adopt the DOLCE ontology. The next section begins with an introduction to DOLCE and proceeds with the alignment of ADACOR to this ontology. Then, we show how to formalize (a part of) a crucial example. Section 6 concludes with some general remarks.

2. Manufacturing Problem Description

This study applies to manufacturing control systems. We look at a manufacturing enterprise that produces discrete items, and model (part of) the factory plant components as well as aspects of the scheduling, monitoring, and execution processes.

2.1 Manufacturing System Description

The manufacturing enterprises produce products that are offered to the market. The products are described by the product model, which contains all technical data and describes the constitution of a 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 and related information like estimated processing time and requirements necessary to produce the part. An operation is a job to execute and involves one of the following main functions: processing, assembly, storage, transportation, manipulation, maintenance or inspection. Each operation has aggregated a set of services.

A customer interacts with a company to order one of the available products or a new product. This order, known as customer order, involves the reference to a product, a quantity, a deliver date and a price. Additionally, it is necessary to create forecast orders to anticipate the market demands. The manufacturing planning convert the customer and forecast orders into production orders, aggregating if possible several customer orders into a production order, to obtain volume and transport advantages. The production orders must specify a quantity, a delivery date and a cost. A production order is indexed to a product object and comprises a list of work orders. A work order is a job that should be executed by a resource.

The shop floor consists of a group of resources (such as movers, transporters, drilling, milling, and turning machines) with different characteristics (spindle speed, list of tools and grippers, tool length compensation, payload, time autonomy, etc.), which have to be carefully described in the factory model. 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 availability of a resource is represented by an agenda that indicates the list of orders allocated to the resource over the time. The agenda comprises also time slots where the resource is: free, allocated to execute orders, temporarily out of service (due for example to maintenance) and out of service.

2.2 Manufacturing Control Description

The main functions required by a manufacturing control system are process planning, resource allocation planning, plan execution, and pathological state handling.

The production of a product involves the execution of several steps, according to a precedence diagram, defined in the process plan. At the process planning level, the manufacturing orders are launched to the shop floor, associated to a process plan that defines 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, each one indicating the exact resource that should execute each operation.

The resource allocation planning 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, in order to produce the products, minimizing the costs and increasing the productivity, and organizing the production unit to react to any modification in demand or machine failure.

The plan execution functions deals with the physical implementation of the schedule into the factory through the dispatching of the scheduled orders to the manufacturing process, and with the production progress monitoring. The reaction to disturbances is initially taken by the execution plan level, and may imply the need of rescheduling of the operations with the aim of minimizing the effects of the disturbance.

2.3 Towards a Manufacturing Ontology

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.

In the communication between distributed and autonomous entities, besides the issues related to interfaces and protocols, it is important to verify that the semantic content is preserved during the exchange of messages. These distributed entities need to have a common understanding of the concepts of their domain knowledge, which is given by a domain (or core) ontology [7]. The inter-operability in distributed and different multi-agent or holonic platforms increases the need for shared ontologies, in order to allow the exchange of knowledge between those distributed platforms.

3. ADACOR Holonic Control Architecture

Our work concentrates on an application domain where several different approaches are implemented and improved continuously. To ground the discussion, we must first

select one architecture and one foundational ontology and then provide an ontological assessment of the concepts adopted by the first through the knowledge structure provided by the latter. Once the notions of this architecture have been ontologically analyzed and classified, we can use the resulting system as a core ontology in the manufacturing domain, perhaps including new concepts from other architectures.

One of the proposed architecture for the manufacturing control is ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) [8], which addresses the agile reaction to disturbances at the shop floor level, increasing the agility and flexibility of the enterprise, when it works in volatile environments, characterized by the frequent occurrence of unexpected disturbances. In the following sections, we introduce this system and clarify its ontological stand.

3.1 Overview of ADACOR Manufacturing Control System

The ADACOR architecture is based in the Holonic Manufacturing Systems (HMS) paradigm[3], 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.). A generic ADACOR holon comprises the Logical Control Device (LCD) and the physical resource capable of performing the manufacturing tasks, if it exists. The LCD device is responsible for regulating the logic activities related to the holon and comprises three main components: decision, communication and physical interface components [8].

The ADACOR architecture groups the manufacturing holons into product, task, operational and supervisor holon classes. Each available product to be produced in the factory plant is represented by a product holon that contains all knowledge related to the product and is responsible for the short-term process planning. Each production order launched to the shop floor in order to execute a product (or sub-product) is represented by a task holon, which is responsible to manage the execution, containing the dynamic information about the production order. Operational holons represent the physical resources available in the shop floor, such as operators, robots and numerical control machines, managing their behaviors according to the resource goals and skills. The supervisor holon introduces coordination and global optimization in decentralized control approaches and is responsible for the group formation and coordination.

The ADACOR adaptive production control balances between a more centralized and a more flat approach, due to the self-organization associated to each ADACOR holon, translated in the autonomy factor and in the propagation mechanisms.

---

[3] HMS (http://hms.ifw.uni-hannover.de/) translates to the manufacturing world the concepts developed by Arthur Koestler for living organisms and social organizations [9]. Holonic manufacturing is characterized by holarchies of holons (i.e., autonomous and cooperative entities), which represent the entire range of manufacturing entities. A holon, as Koestler devised the term, is a part of a (manufacturing) system that has a unique identifier, may be made up of sub-ordinate parts and, in turn, can be part of a larger whole.

3.2 The ADACOR Manufacturing Ontology

ADACOR defines its own manufacturing ontology, expressed in an object-oriented frame-based manner as recommended in the FIPA Ontology Service Recommendations [10]. Thus, the architecture uses classes to describe concepts and predicates and fixes them as part of the application ontology. This allows for a practical and fast way of creating an ontology with an immediate underlying implementation.

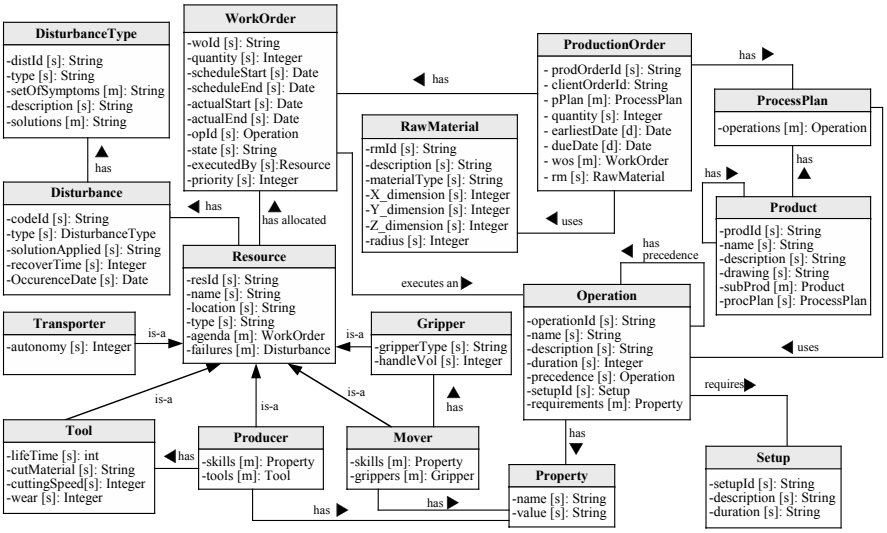

Fig. 1. Manufacturing Ontology Developed in the ADACOR Architecture

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-like diagram of Figure 1). 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, since a production order may index one customer order or an aggregation of these, this relationship and the latter concept are not shown.

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 (see 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 bolds (unless produced internally).

- *Customer order*: entity that the enterprise receives from a customer that requests some products.
- *Production order*: entity obtained by converting the customer and forecast orders (it may result from the aggregation of several customer orders).
- *Work order*: entity generated by the enterprise in order to describe the production of a product. The work order lists one or more operations including their processing time, participants (e.g. name and number of resources involved in the execution), priority, scheduled dates, state and quantity.
- *Resource*: entity that can execute a certain range of jobs as long as its capacity is not exceeded. Producer, mover, transporter, tool, and gripper are specializations of the resource object and inherit its characteristics[4].
- *Operation*: a job executed by one resource. There are different types of operations among which drilling, maintenance, and reconfiguration of resources.
- *Disturbance*: unexpected event, such as machine failure or delay, that degrades the original production plan.
- *Process Plan*: description of a sequence of operations, including temporal constraints like precedence of execution, for producing a product.
- *Property*: an attribute that characterizes a resource or that a resource should satisfy to execute an operation.

Predicates are expressions that allow to establish relationships among concepts. The main predicates in the ADACOR ontology are informally described as follows:

- *SubproductOf(x,y)*: *x* is a product which is a sub-product (a component) of *y*.
- *Allocated(x,y,i)*: operation *x* is allocated to resource *y* during time interval *i*.
- *Available(x,y,t)*: resource *x* is available at time *t* to perform operation *y*.
- *RequiresTool(x,y)*: operation *x* requires tool *y*.
- *HasTool(x,y,t)*: resource *x* has tool *y* available in its tool magazine at time *t*.
- *RequiresSkill(x,y)*: execution of operation *x* requires property (skill) *y*.
- *HasSkill(x,y)*: resource *x* has property (skill) *y*.
- *HasFailure(x,y,t)*: a disturbance *x* occurred in resource *y* at time *t*.
- *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*.

4. Interoperability in Manufacturing Control Applications

The ontologies currently used in the manufacturing domain are the result of non-coordinated efforts and relinquish the interoperability with other agents communities. As seen in section 3.2, in the ADACOR architecture a basic and proprietary manufacturing ontology has been developed to support the inter-operability between ADACOR holons. However, the lack of inter-operability 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.

[4] Here we do not consider human operators which, for completeness, should be listed among the resources of the system. Indeed, sometimes operations like maintenance or reconfiguring must be executed by human operators.

Lately, several efforts to develop standard mechanisms for the unambiguous exchange of information within the manufacturing domain have been undertaken. The International Organization for Standardization (ISO) developed STEP (Standard for the Exchange of Product Model Data) that defines a standard data format for exchanging a complete product specification (e.g. geometry and production process) between heterogeneous CAD/CAM systems. However, STEP refers to the product information only and does not cover the process and enterprise engineering information. A set of initiatives seeks to fulfill this gap. The Process Specification Language (PSL) project [11] aims to develop 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) [12] identifies information models to facilitate process specification and to transfer this information to process control. The Toronto Virtual Enterprise (TOVE) [13] defines a domain-specific ontology for enterprise modelling. 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" [14]. The goal of the Process Interchange Format (PIF) project [15] is to support the exchange of business process models across different formats and schemas. We conclude with the Plinius project [16], whose goal is to define a domain-specific ontology for mechanical properties of ceramic material. Of course, this list of projects is far from complete, it is provided just to show the variety of approaches and standardization initiatives in this area.

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. The application of foundational ontologies to support the interoperability between agent-based and holonic manufacturing control applications provides a feasible and reliable way to solve this problematic situation. Also, 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.

Just a few foundational ontologies have been developed and motivated to a satisfactory level in the literature, in particular DOLCE (the Descriptive Ontology for Linguistic and Cognitive Engineering, http://www.loa-cnr.it/Ontologies.html), GFO (the General Formal Ontology [17], http://www.onto-med.de), OCHRE (the Object-Centered High-level Reference Ontology, developed by L. Schneider [3]), OpenCyc (http://www.opencyc.com), SUMO (the Suggested Upper Merged Ontology [18], http://www.ontologyportal.org) and, although only partially formalized, BFO (the Basic Formal Ontology. developed at IFOMIS [3], http://www.ifomis.uni-leipzig.de).

Since foundational ontologies are complex systems, there are two crucial elements that should be considered in choosing an ontology: the ontology has to provide a rich set of conceptual distinctions (at least relatively to the domain of application), and all the features that one deems relevant should be clearly characterized (or characterizable) within the ontology. In our case, the chosen foundational ontology is the DOLCE ontology because it distinguishes 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 proper-

ties and capacities, and it relies on a very expressive language, namely first-order modal logic[5]; all features crucial in modelling physical objects, agents, and processes. Even more so, DOLCE let the user define the qualities needed in the application, allowing in this way a great level of freedom while facilitating update and maintenance. Finally, as said in the introduction, the application of a foundational ontology should be coupled with a lightweight version of that very ontology: lightweight versions of DOLCE are available in LOOM, DAML+OIL, RDFS, DIG, and OWL.

5. Formalization of the ADACOR Ontology in DOLCE

DOLCE, the foundational ontology developed at the Laboratory for Applied Ontology (ISTC-CNR, http://www.loa-cnr.it), is mainly an ontology of particulars in the sense that it focuses on this class of entities. Universals (predicates) are considered in so far as they help in the classification of particulars. This ontology adopts the multiplicative approach, that is, it assumes that different entities can be co-located in the same space-time. Co-located entities differ because they enjoy incompatible properties, for example, a drilling machine does not survive a radical shape deformation while its amount of matter does, therefore the machine and the amount of matter are different entities in DOLCE, yet co-located. An important aspect of DOLCE is the treatment of qualities. Endurants (objects like a gripper, a person) and perdurants (events like making a hole, moving a steel block) come with a bunch of qualities, e.g. shape, weight, duration, velocity, etc. Qualities may be specific to a subclass of entity, for instance weight is a quality of physical endurants only. An entity like Hammer_#123 has its own individual qualities: its shape, its weight, its color, etc. that exist as long as that hammer does. These individual qualities are elements in DOLCE so that one can refer to them directly in formal expressions. For each quality, there exists a quality space: the quality space of shape, the quality space of weight, etc. Each individual quality of an entity (say, its weight-quality) is associated to a position in the corresponding quality space (the weight-space) and this position is called its quale. This allows us to make important distinctions and comparisons even *before* introducing measurement. Indeed, measures depend on units of measurement and methodologies, thus are obtained only once these elements have been fixed. Independently of the measurement, at each point in time the hammer weight-quale is a precise position in the weight quality space. Two hammers that have the same weight-quale, must have the same weight-measure, no matter how we assign measures to positions in the space.

On a different level, the DOLCE ontology has been compared to other foundational ontologies (e.g. OCHRE and BFO in [3]) and it is included in other merging initiatives [19]. This is important to our project: it is generally granted that interoperability is obtained through the compliance with ontologies thus, if DOLCE is included in merging initiatives, our core ontology is likely to be easily linked to other manufacturing ontologies, at least those developed for interoperability.

---

[5] This should not be surprising. Foundational ontologies are used to structure the knowledge base and are not applied at run-time when computability and effectiveness issues are crucial.

The taxonomy of the most basic categories of particulars in DOLCE is depicted in Figure 2. An informal (and partial) description of the main predicates is given next. We refer the reader to [3] for the formal characterization of these predicates and a throughout discussion of the DOLCE assumptions.

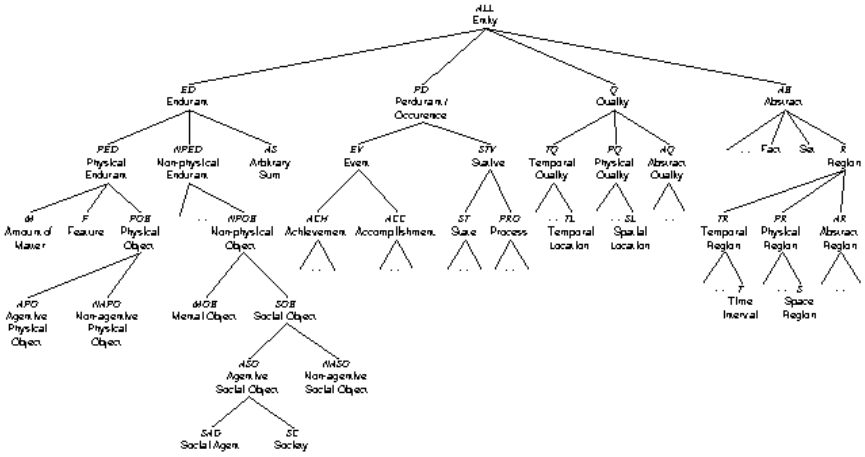

Fig. 2. Taxonomy of DOLCE basic categories [3]

ED(x), PED(x) stand for "*x is an endurant*" and "*x is a physical endurant*", respectively. An endurant is an entity that is *wholly* present at any time it is present. It is physical if located in space and time: a hammer, a mover machine, an amount of plastic. See the predicate NASO below for examples of non-physical endurants.

PD(x) stands for "*x is a perdurant or event*", i.e., an entity that is only partially present at any time it is present. For instance, consider the perdurant "producing an item of type #234" that consists in attaching two metal pieces together with screws and painting the resulting piece. While the painting goes on, the (temporal) part corresponding to attaching the two pieces is not present anymore and when this is present, the painting is not yet. Perdurants can have spatial parts as well. Note that objects are not parts of perdurants, rather objects participate in them. Perdurants form four sub-classes: achievements, accomplishments, states, and processes.

In the manufacturing domain, one needs to refer to a wide variety of operations. Some of these, like turning, are said to be homeomeric, i.e., every part of a turning is a turning itself. In other terms, if one divides a turning operation with temporal interval t in two parts, one initial operation with interval t_1, and one final operation with interval t_2 (such that t_1 and t_2 partition t), then both the initial and the final operations have all the characteristics of a turning operation. This does not happen with, say, a setup operation. It is necessary for a setup to reach a specific state since it is the achievement of such a state that justifies its classification as a setup. Thus, if we divide a setup operation in two temporal parts, only one of the two sub-operations (if any) can be considered a setup operation. This and similar distinctions will drive the ontologi-

cal classification of the ADACOR notions and are captured by the DOLCE predicates below.

ACH(x) stands for "*x is an achievement*". These perdurants are characterized by anti-cumulativeness (the sum of two achievements of, say, type A is not an event of type A) and atomicity (they do not have temporal parts). E.g., the completion of a machine reconfiguration is an achievement but the reconfiguration itself is not.

ACC(x) stands for "*x is an accomplishment*". These are non-atomic perdurants, i.e., they have temporal parts. For example, a reconfiguration can be composed of several sub-events (like the reconfiguration of different parts of a production line). However, the sum of reconfigurations of some type A is never a reconfiguration of the same type, that is, accomplishments enjoy anti-cumulativeness.

ST(x) stands for "*x is a state*". This class of perdurants is cumulative, thus it is closed under mereological sum in the sense that the sum of two perdurants of type A (say, drilling events) is a perdurant of the same type (a drilling). Also, these perdurants are homeomeric. Drilling and moving perdurants are in this class.

NAPO(x) stands for "*x is a non-agentive physical object*". These are objects that have spatial and temporal location but to which one cannot ascribe intentions, believes, or desires; e.g., products and production orders.

NASO(x) stands for "*x is a non-agentive social object*". These objects have neither (direct) spatial or temporal location nor intentions, believes, or desires. They depend generically on societies like laws and plans.

qt(q,x) stands for "*q is a quality of x*". Qualities are basic 'properties' of entities. They can be perceived or measured. In this sense, they represent partial characterizations of an entity and depend existentially on it. Every endurant (perdurant) comes with its physical (temporal) qualities.

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. The quale is the position of an individual quality in the corresponding quality space. If a quality space is poor, then there are few different positions in it and it is likely that corresponding individual qualities (of different entities) are associated to the same quale. Rich quality spaces allow for finer distinctions among qualities. Two entities with individual qualities associated to the same quale (in their quality space) are undistinguishable regarding to the corresponding quality.

5.1 Alignment of ADACOR with DOLCE

DOLCE provides a natural way to classify entities in the ADACOR architecture. Beside the basic distinction between endurants and perdurants, descriptions can be modelled explicitly as a type of objects, and properties are simply qualities. Below, we present the classification of some entities of section 3 according to our work.

Products, resources and orders[6] are physical, non-agentive objects (NAPO)

$$(\text{Product}(x) \geq \text{Resource}(x) \geq \text{Order}(x)) \geq \text{NAPO}(x)$$

[6] The notions of resource and agent are related. Section 5.3 discusses these in the manufacturing domain. Also, note that we distinguish between orders and order-descriptions.

In ADACOR, raw-material refers to objects and to amounts of matter as well, thus we classify raw-material as generic physical endurants (PED)

Raw_material(x) ≥ PED(x)

At this point, we constrain the meaning of the terms "Order" and "Resource" in ADACOR, that is, we formalize the predicates of ADACOR that are new in DOLCE,

Order(x) ≥ (Production_order(x) ≥ Customer_order(x) ≥ Work_order(x))

Resource(x) ≥ (Producer(x) ≥ Mover(x) ≥ Transporter(x) ≥ Gripper(x) ≥ Tool(x))

The remaining entities of section 3 are perdurants (PD) since they identify activities or states. DOLCE makes a clear-cut distinction between achievements (ACH), accomplishments (ACC), states (ST), and processes (PRO). We found this partition of perdurants very helpful and it is used systematically in the system. Note that "Operation" and "Disturbance" are disjoint top classes of ADACOR perdurants[7].

Operation(x) ≥ PD(x)

Disturbance(x) ≥ ACH(x)

¬ (Operation(x) ≥ Disturbance(x))

Completion(x) ≥ ACH(x)

(Setup(x) ≥ Reconfiguration(x) ≥ Inspection(x) ≥ Maintenance(x) ≥ Assembly(x) ≥ Production(x)) ≥ ACC(x)

(Transportation(x) ≥ Turning(x) ≥ Drilling(x) ≥ Milling(x)) ≥ ST(x)

Most notably, in our limited list we find no instance of the DOLCE notion of process. Consider, for instance, transportation: since all the temporal parts of transportation are transportation as well, this type of event is stative (ST). A similar argument holds for turning, drilling, and milling.

As done for the endurants, we must characterize the meaning of the general terms "Operation", "Disturbance", "Completion" and "Reconfiguration" in our formalization. The constraints are given below.

Operation(x) ≥ (Completion(x) ≥ Reconfiguration(x) ≥ Inspection(x) ≥ Setup(x) ≥ Maintenance(x) ≥ Assembly(x) ≥ Production(x) ≥ Milling(x) ≥ Transportation(x) ≥ Turning(x) ≥ Drilling(x))

Disturbance(x) ≥ (Failure(x) ≥ Delay(x))

Completion(x) ≥ (Completion_of_setup(x) ≥ Completion_of_inspection(x) ≥ Completion_of_assembly(x) ≥ Completion_of_reconfiguration(x) ≥ Completion_of_maintenance(x) ≥ Completion_of_production(x))

Reconfiguration(x) ≥ (Addition_of_new_resource(x) ≥ Change_of_layout(x) ≥ Removal_of_resource(x) ≥ Change_of_resource_capability (x))

---

[7] In principle, one can consider disturbances as operations. However, this seems unnatural to people working in manufacturing. For this reason, disturbances and operations are presented as disjoint classes.

There is a misalignment between the notion of Setup as an operation (above) and the concept of Setup in Figure 1. Some operations may have a Setup operation as requirement and this is the reason to show Setup as a separate entry in Figure 1.

The notions of Delay, Disturbance, and Failure are related concepts and their formalization require some caution. A disturbance is an unexpected event: machine failure or machine delay are the only examples of disturbances we consider. These are crucial since they affect the scheduled production plan. When an operation is being executed, we can expect several different scenarios: (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, Failure is a kind of accomplishment. However, the classification of Delay is less obvious: a delay occurs when the need of rescheduling is officially established, thus it is an atomic event. Also, the sum of two delays is not a delay since the sum does not correspond to a single rescheduling requirement. Thus, Delay is taken to be an accomplishment as well.

As mentioned in the manufacturing description, a "Process Plan" is a description of a sequence of operations (plus related properties and interconnections). Then, a Process Plan is a non-agentive social object (NASO), which implies that it is non-physical. Indeed, we distinguish the description (a non–physical object) from the document that contains the description (a physical object)[8]. Here it suffices to consider the first:

$$\text{Process\_plan}(x) \geq \text{NASO}(x)$$

In the terminology of DOLCE, skills are qualities of objects. For each type of skill we must include a quality space and, for each object that has that skill, an individual quality specific to that object. It is crucial to note that we inherit from DOLCE the distinction between individual qualities (stating that the object at stake has that skill) and the corresponding quale (roughly, a classification of the object's skill). The quale shows the characteristic of that object with respect to the given skill. For example, consider "Autonomy" to be an individual quality[9] enjoyed by *any* resource. It characterizes for how long a resource can work without the need to re-fill its batteries (assuming a way of measuring this skill is given). If AutL is the class of autonomy-qualities, then the following constraint states that "Autonomy" is a quality defined for resources only

$$\text{AutL}(q) \geq \geq x\ (\text{qt}(q,x) \geq \text{Resource}(x))$$

The specific relations "q is the autonomy of resource x" and "resource x has autonomy d at time t" are not part of the language but can be defined in it as follows

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

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

---

[8] This distinction raises in different forms and it is pervasive. For instance, one should distinguish between order and order-description, operation and operation-description, and so on. This issue is related to the notion of role. See [20] for a treatment of roles in DOLCE.

[9] One could take 'Autonomy' to be reducible to other simpler qualities. This alternative view is compatible with the characterization we provide.

Every resource must be explicitly associated to a (finite) set of qualities or skills that capture its characteristics, e.g. Autonomy, Magazine_capacity, Max_feed_rate. If $QL_1,\ldots,QL_n$ are the skills of resource A_i, then we set the following constraint

$$\geq q_1,\ldots,q_n(QL_1(q_1,A_i) \geq \ldots \geq QL_n(q_n,A_i))$$

Then, by using skill indices in the argument we can define the relation Has_skill

$$\text{Has\_skill}(y,j) =_{def} \text{Resource}(y) \geq \geq q\ QL_j(q,y)$$

Sometimes we must be able to state some general condition (like "resource *x* has tool *y* available") or to select resources that not only have a given skill but that can perform it in a certain way. These cases are captured through the notion of "Requirement", that is, through relations like "Has_drilling_feed_speed_rate", "Has_tool", etc., that describe some general properties. These relations are often defined by complex logical expressions so here some of them are presented with a minimal characterization only (below we use QL_d for a quality related to drilling, this is not characterized further; f-s-r stands for 'feed speed rate')

$$\text{Executes}(x,y,t) \geq (\text{Resource}(x) \geq \text{Operation}(y) \geq T(t))$$
(resource x executes operation y at time t)

$$\text{Has\_tool}(x,y,t) \geq (\text{Resource}(x) \geq \text{Tool}(y) \geq T(t))$$
(resource x has tool y available at time t)

$$\text{Has\_drilling\_feed\_speed\_rate}(x,t,a,b) =_{def} (\text{Resource}(x) \geq \geq q\ (QL_d(q,x) \geq \geq y\ ql(y,q,t) \geq a \leq y \leq b)$$
(resource x has drilling f-s-r between a and b at t)

$$\text{Requires\_skill}(x,j) =_{def} (\text{Operation}(x) \geq \geq y,t\ (\text{Executes}(y,x,t) \geq \text{Has\_skill}(y,j)))$$
(operation x requires skill y to be executed)

$$\text{Setup\_requires\_tool}(x,y) =_{def} (\text{Setup}(x) \geq \geq z,t\ (\text{Executes}(z,x,t) \geq \text{Has\_tool}(z,y,t)))$$
(setup x requires tool y to be performed)

$$\text{Milling\_requires\_autonomy}(x,t,y) =_{def} (\text{Milling}(x) \geq \geq z,d\ (\text{Executes}(z,x,t) \geq \text{Autonomy}(d,z,t) \geq d \geq y))$$
(milling x at time t requires autonomy at least y)

5.2 An Example: Bidding for a Job Task

We concentrate on a specific example, an instance of a task allocation interaction, and show how the ontology shapes its formalization in the system. Here, we limit ourselves to the language fragment introduced in sections 3 and 5.

The agent **t1** (contractor) has a job task that comprises two different operations, "mach-piece" and "drill-holes". The first operation has precedence over the second, that is, the "drill-holes" operation can start only once the first operation has been completed. The "mach-piece" operation for this job must be executed by a resource with the following characteristics: it should be a milling machine with feed speed 1000. For the second operation the following is needed: it should be executed by a drilling machine with the feed speed 700. Agent **t1** sends a message to all agents connected to the system. The message announces an operation and the requirements for it. For example, the message for the operation "drill-holes" is:

```
(Cfp
 :sender (agent-identifier :name t1)
 :receiver (agent-identifier :name mach-a, mach-b)
 :language FIPA-SL0
 :ontology Adacor-ontology
 :protocol FIPA-Contract-Net
 :content ((ONLY-OPERATION (OPERATION :name drill-holes
    :exectime 55 :rawmaterial steel-100 :precedence mach-piece
    :properties (set (PROPERTY :name mach_type :value drilling)
    (PROPERTY :name speed :value 700)) :quantity 100 :state
    NOT-ALLOCATED :earlieststart 094248884 :duedate 094316884))
)
```

The message contains several fields where the language, ontology and protocol used in the message construction are reported. The content of the message stores the information that the contractor wants to send to the contractees. In this case the content has information formatted using the "Operation" concept.

Without entering into the details of the message configuration and exchange protocols, our ontological assessment of the terminology allows us to share information with a clear meaning (through the formal semantics of the DOLCE ontology) by including logical expressions in the message. Let us write "DrillH1" for this specific operation, then the entry content is explicitly characterized by the following[10]:

Bidder(x,DrillH1,t) ≥

(Resource(x) ≥

≥y (Has_tool(x,y,t) ≥ Drilling_tool(y)) ≥ Available(x,DrillH1,t) ≥

≥ j (Requires_skill(DrillH1,j) ≥ Has_skill(x,j)) ≥

≥ a,b (Has_drilling_feed_speed_range(x,t,a,b) ≥ a≤700≤b) ≥
Autonomy(d,x,t) ≥ d≥55)

where "Bidder(x,DrillH1,t)" stands for "*x bids to perform operation DrillH1 at time t*", and "Drilling_tool(y)" is a specialization of "Tool(y)".

With the alignment of ADACOR to DOLCE, the above logical expression states formally (in an explicit and ontologically sound setting) that if an entity x bids for the job DrillH1 at a time t, then x has the following (now unambiguous) characteristics:

- x is a resource
- a tool to execute drilling operations is available to x at time t
- x is available at time t to execute the drilling operation DrillH1
- x has all the skills required by the drilling operation DrillH1
- x has capacity to drill at 700 feed speed rate at time t
- x is autonomous for at least 55 time-units

Each resource agent verifies its capabilities to execute the operation (both in terms of skills and calendar) and, if it finds a time t such that the conditions above are satisfied, it answers the call for operation DrillH1 and time t.

[10] This is only a partial characterization limited to the adopted language fragment.

If agent **mach-a** wants to bid the announcement, it produces this message:

```
(Propose
 :sender (agent-identifier :name mach-a)
 :receiver (agent-identifier :name t1)
 :language FIPA-SL0
 :ontology Adacor-ontology
 :protocol FIPA-Contract-Net
 :content ((OP-PROPOSE (OPERATION :name drill-holes :exectime
   55 :rawmaterial steel-100 :precedence mach-piece :properties
   (set (PROPERTY :name mach_type :value drilling) (PROPERTY
   :name speed :value 700)) :quantity 100 :state NOT-ALLOCATED
   :earlieststart 094248884 :duedate 094316884)(PROPOSAL
   :name mach-a :price 100 :location IDIT)))
)
```

The bid message is similar to the first one but now the content of the message contains different information, translated with the relation between the "Operation" and the "Proposal" concepts. The proposal data structure comprises the name of the entity that is sending the proposal, its physical location and the price proposed to execute the operation. In particular, if the system is formalized according to the alignment to the DOLCE ontology, agent **mach-**a can send a logical expression stating that this agent can execute the operation DrillH1, that it satisfies all the constraints (this part is obtained easily from the one given above), and other relevant information like restrictions in the operational skills that may interfere with the job execution. Furthermore, the message will include a new piece of information[11]

Proposal(**mach-a**,**t1**,DrillH1,100,IDIT)

where the predicate "Proposal", being ontologically characterized, is now a formal concept with clear meaning and implications (obligations, responsibilities, legal rights) even for the new entities not previously connected to the system.

5.3 Issues in Modelling the Manufacturing Domain

If the need to distinguish between values (essentially, numbers) and value ranges can be taken for granted in the manufacturing domain, subtler distinctions are not. We find helpful to distinguish a skill of a machine (for instance, being able to execute a particular job like drilling a hole) from qualitative and quantitative aspects of that skill (the way the hole is obtained, the speed of execution and so on). Indeed, being able to execute a particular operation is a necessary condition to answer a bid for a work order independently of qualitative and quantitative aspects. On the other hand, these aspects are necessary for any rescheduling process. Similar distinctions arise in dealing with time. Talking of the expected duration of an operation, one may need to refer to the precise event (the operation executed by a given resource at a given time), the duration of that event (the time is spans), and the length of a temporal interval.

A different set of problems raises from the different conceptualization of the entities in the domain. For instance, the notion of agent in the manufacturing community

[11] For the sake of the example, here we assume that a single operation can play the role of a work order.

is often application-dependent, that is, the very same entity might (or might not) play the role of an agent in a manufacturing process depending on the application we are considering. However, in other cases the notion of agent is used in absolute terms, that is, an agent is any entity that has the capacity to initiate actions (perhaps in a proactive and rational way). An analogous argument holds for the notion of resource as used in this paper. These views appear to be incompatible since in the first case simple tools (like a hammer) might be considered on a par with machines (complex floor resources) while in the latter case such tools are ontologically distinct from agents. In order to be transparent across different ontologies, the modeller must take this different conceptualizations seriously. For instance, one can classify these entities as (generic) physical objects and allow the application specifications to introduce further differentiations among the entities (with corresponding ontological properties attached to them). This issue could be solved within DOLCE exploiting the use of quality spaces or introducing roles explicitly.

Other entities seem hard to formalize because we point to different aspects in different contexts. An example is the notion of raw-material. Consider a company that produces clothes and that buys buttons from another producer. For the company, the buttons are classified as raw-material. Indeed, this company conceptualizes buttons as "components" of the items it produces and not as "products" themselves. However, the very same items are products for the button producer. This discrepancy is only apparent since ontologically the items we refer to (talking of raw-material or product) have the same properties in all contexts; they are physical objects with well defined characteristics. The real problem is that raw-material is not an ontological distinction and thus it collects things of different ontological types like amounts of matter (sand, water, steel and the like) and complex artefacts (buttons, pipes, hammers). This is the reason we classify them as generic (physical) endurants. Further characterizations have to take into account the specific items one is talking about.

6. Conclusions

Our work, here presented only in part, aims at a domain-specific ontology (core ontology) for the manufacturing production field. Once completed, the resulting ontology will be well-founded, in particular because driven by a foundational ontology. Also, it inherits the advantages of a richly characterized ontology making it suitable for information communication and sharing. On the other hand, it is modelled by the subject field because specific applicative concerns have driven the alignment and refinement of the initial vocabulary. In short, the combination of a foundational ontology and an application architecture to produce a core ontology has the advantage of mixing bottom-up and top-down strategies maintaining crucial characteristics of both.

As of today, the concrete application of foundational ontologies is rather limited. We believe this situation is due more to structural than to scientific reasons. The development of foundational ontologies requires highly interdisciplinary efforts and involves expertise in a variety of areas (logic, philosophy, linguistics, conceptual modelling, information systems). There are only a few research groups that cover adequately these areas and that are active in ontology for information science. Unfor-

tunately, application domains (enterprise management, medicine, law, and the like) differ considerably so that the application of general ontologies to these domain is necessarily the result of specific collaboration efforts with domain experts. This explains the actual shortage of concrete examples, although recently we have observed increasing interests in the exploitation and comparison of application experiences centred upon foundational ontologies[12]. We believe that when foundational ontologies become available with clear documentation and supporting tools for the non-trained user, we will notice an increasing application of these ontologies and, consequently, an improvement of the average domain-specific ontologies available on the market.

7. References

1. N. Guarino, Formal Ontology and Information Systems, FOIS 1998, Trento, Italy, pp 3-15.
2. C. Fellbaum (ed.), WordNet An Electronic Lexical Database, Bradford Book, 2000.
3. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, Ontology Library (WonderWeb Deliverable D18), http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf
4. D.M. Pisanelli, A. Gangemi, G. Steve, Ontologies and Information Systems: the Marriage of the Century?, Proceedings of Lyee Workshop, Paris 2002.
5. P. Bertolazzi, C. Krusich, M. Missikoff, An Approach to the Definition of a Core Enterprise Ontology: CEO, OES-SEO '01, Rome, 2001.
6. Y. Wilks, Ontotherapy: or how to stop worrying about what there is, 2003/8/6 http://www.racai.ro/EUROLAN-2003/html/presentations/SheffieldWilksBrewsterDingli/Ontotherapy_YWilks.ppt
7. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, Understanding Top-Level Ontological Distinctions, Workshop on Ontologies and Information Sharing, IJCAI 2001.
8. P. Leitão and F. Restivo, Holonic Adaptive Production Control Systems, Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society, 2002, pp 2968-2973.
9. A. Koestler, The Ghost in the Machine, Arkana Books, London, 1969.
10. Foundation for Intelligent Physical Agents, http://www.fipa.org/ (June 2003).
11. C. Schlenoff, A. Knutilla, S. Ray, Unified Process Specification Language: Requirements for Modelling Process, NIST, Interagency Report 5910, Gaithersburg MD, September 1996.
12. B. Catron, S. Ray, ALPS: A Language for Process Specification, International Journal of Computer Integrated Manufacturing, Vol. 4, No. 2, 105-113, 1991.
13. F. Fadel, M. Fox and M. Gruninger, A Generic Enterprise Resource Ontology, 3rd IEEE Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, 1994.
14. M. Uschold, M. King, S. Moralee, Y. Zorgios, The Enterprise Ontology, The Knowledge Engineering Review, 13(1), Special Issue on Putting Ontologies to Use, pp. 31-89, 1998.
15. J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, G. Yost and the PIF WG, The PIF Process Interchange Format and Framework, Know. Eng. Rev. 13(1), pp.91-120, 1998.
16. P. van der Vet, P.-H. Speel, N. Mars, The PLINIUS Ontology of Ceramic Materials. ECAI'94, Workshop on Comparison of Implemented Ontologies, 1994.
17. B. Heller, H. Herre, Ontological Categories in GOL, Axiomathes (14):1 pp 57-76.
18. I. Niles and A. Pease, Toward a Standard Upper Ontology, FOIS 2001, pp. 2-9
19. P. Martin, Correction and Extension of WordNet 1.7, LNAI 2746, pp 160-173.
20. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N., Social Roles and their Descriptions, KR 2004, pp. 267-277

[12] This is one motivation for the workshop on "Core Ontologies in Ontology Engineering" which is held at EKAW 2004, http://www.loa-cnr.it/core_onto.html