Formal Ontology in Information Systems
F. Neuhaus and B. Brodaric (Eds.)

doi:10.3233/FAIA210368

Towards Formalisation of Concept Descriptions and Constraints

Matt SELWAY [1], Markus STUMPTNER and Wolfgang MAYER
Industrial AI Research Centre, UniSA STEM,
University of South Australia, Australia

Abstract. The integrated management of industrial systems in future environments like Industry 4.0 requires the effective management of information throughout the engineering life cycle. As systems pass through phases of design, construction, operation, maintenance, renewal or replacement, they will be administered via different information ecosystems, requiring changing perspectives on their descriptive information. A central role in the interplay of software and hardware artefacts, functions, documentation and managing software is played by the *description*s of concepts (i.e. formalised definitions of concepts within the domain of quantification). In this paper we propose a unified formalisation of *description*s that permits consistent analysis of the relationships between the designs, types, products, and concrete artefacts that can be found in the industrial engineering life-cycle. The approach is consistent with our earlier framework that describes artefacts, requirements and functional roles in the context of the DOLCE foundational ontology.

Keywords. Interoperability, Artefacts, Descriptions, Relationships

1. Introduction

Effective construction and operation of industrial plants requires the management of extensive bodies of information about the plant and its surrounding operational and maintenance activities across the entire life cycle of the plant. Figure 1 depicts the life cycle of a particular "object" in engineering parlance, such as a pump that is part of a vehicle or an industrial plant. The pump itself is a complex object composed from multiple parts—with its specification, design information, and maintenance/fault records reflecting this subdivision—while being itself only a small part of the whole plant. Each stage, from requirements through specification, design, construction, operation, and maintenance, may be subject to information being handled by different software systems based on different languages, data models, or assumptions concerning scope and interpretation of data. As a result, establishing and sustaining interoperability between these heterogeneous systems is a long standing challenge, in particular in heavy industry sectors.

[1]Corresponding Author: Matt Selway, University of South Australia, GPO Box 2471, Adelaide SA 5001, Australia; E-mail: matt.selway@unisa.edu.au.

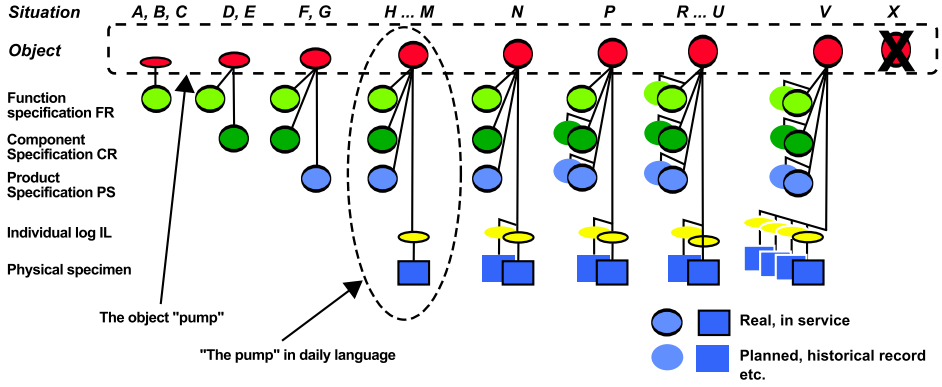

Figure 1. Engineering Life Cycle of a Pump Object (Source: POSC Caesar Association)

Information modelling and exchange standards have been developed to facilitate sharing of information among these information systems, but they remain generally confined to particular subsections of the life cycle. Ontology based information modelling offers to help overcome the information silos by providing a principled view of the problem domain and the associated information artefacts. Such an overarching reference model can then be used to mediate between the individual system's information models by expressing each individual information system in terms of the reference model and exploiting the resulting mappings in the transformations that implement the information exchange protocols. For the construction of suitable transformations and to maintain consistent information across system boundaries, automated validation and integration of information are key. In [1], we presented a framework for such a reference model based on explicit ontologically sound representation of artefacts.

This paper extends the formal coverage of the framework to manage, apply and validate machine interpretable content of the specifications and descriptions associated with an artefact in different life cycle stages, without the development of custom application code for each aspect. In Section 2, we examine the basic requirements for such an explicit representation and introduce declarative *constraints* that embody domain specific invariants on the instances of the ontology that represent a particular collection of technical artefacts relevant to an application. We characterise the properties of constraints and distinguish necessary and obligatory constraints in the context of evolving information. In Section 3 we validate the formal model by applying it to describe the SLICER approach [2] for representation of industrial artefact domains, showing that the framework can restate the core axioms of SLICER within the extended DOLCE ontology from [1].

2. Basic Formulation

In this section we formalise the foundations of our approach. As in [1], the representation is again formalised as extensions to the DOLCE foundational ontology [3] enriched with the notion of *social concept* and *description* introduced in [4], relationships reified in the domain of quantification suggested in [5], artefacts

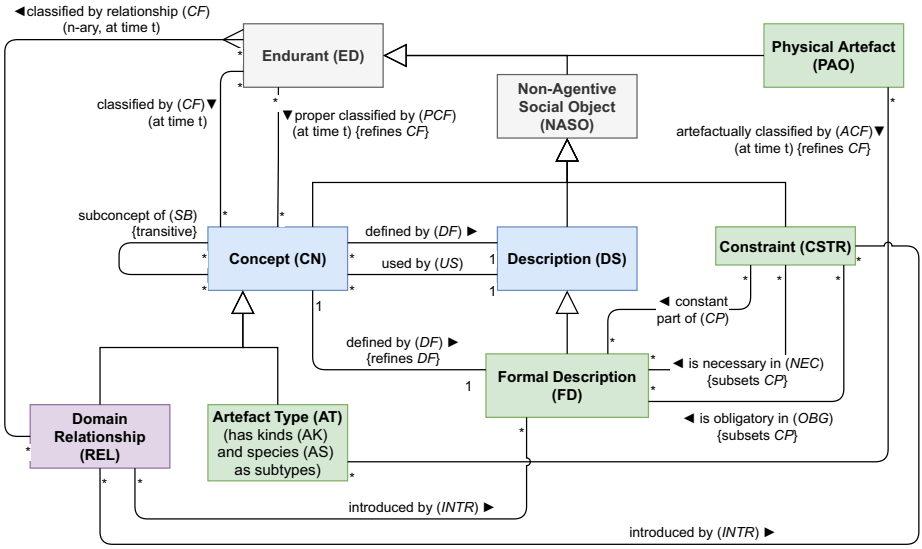

Figure 2. Conceptual overview of the types and relations in the formalisation

proposed in [6], and functions and roles discussed in [7]. For brevity, we consider only a subset of categories to do with concepts and their descriptions. An overview of these concepts is illustrated in Figure 2. In particular, we focus on artefactual objects and their specifications.

(1) $\mathsf{CN}(x) \rightarrow \mathsf{NASO}(x)$ (Concepts are non-agentive social objects)
(2) $\mathsf{REL}(x) \rightarrow \mathsf{CN}(x)$ (Domain Relationships (types) are concepts)
(3) $\mathsf{AT}(x) \rightarrow \mathsf{CN}(x)$ (Artefact Types are concepts)
(4) $\mathsf{AT}(x) \rightarrow \mathsf{AK}(x) \vee \mathsf{AS}(x)$ (Artefact types are A.Kinds or A. Species)[2]

It should be noted that since the basis of our formulation reifies concepts and their descriptions in the domain of quantification, so too must the constraints be reified within the domain of quantification. Therefore, the typical approach that would use, say, modal logic and axioms at the top-level axiomatisation is inappropriate for defining the domain specific constraints. Moreover, the approach taken here, which avoids the use of modal logic, quarantines the domain specific definitions and constraints from the framework used to describe them, allowing local re-evaluation of constraints when data changes in the system (e.g., a new temperature value is sensed). This is important in an information system that may be managing very large sets of instances and constant data updates.

Descriptions (really, *formal* descriptions) uniquely define a concept and comprise constraints (CSTR), which can be *necessary* or *obligatory* (i.e., constraints that can be falsified without impacting an object's classification). This specialises the definition of [4] which allows a description to define multiple concepts or none at all.[3] We take an intensional view that requires concepts to be defined by (*DF*) a

[2]Artefact Kinds are more general types such as 'Pump', while Artefact Species are more detailed, physically and functionally, such as the C12 pump model sold by a manufacturer.

[3]This difference is due to formalising the content of *formal descriptions* to be *constraints* rather than other, simpler descriptions that do not define a concept at all; this does not change

description, indeed it is part of a concept's identity. The constraints extend the content of descriptions to more than the 'used by' relation of [4]. We maintain consistency with [4] in that we consider the semantic content of a description to not change over time (12, 13).

(5) $\mathsf{DS}(x) \rightarrow \mathsf{NASO}(x)$ (Descriptions are non-agentive social objects [4])
(6) $FD(x,y) \rightarrow \mathsf{DS}$ (Formal descriptions are descriptions)
(7) $DF(x,y) \rightarrow \mathsf{CN}(x) \wedge \mathsf{DS}(y)$ (The concept x is defined by the description y)
(8) $\mathsf{FD}(x) \rightarrow \exists! y\, \mathsf{CN}(y) \wedge DF(y,x)$ (9) $\mathsf{CN}(x) \rightarrow \exists! y\, \mathsf{DS}(y) \wedge DF(x,y)$
(10) $\mathsf{CSTR}(x) \rightarrow \mathsf{NASO}(x)$ (A constraint is a non-agentive social object)
(11) $P(x,y,t) \rightarrow \mathsf{ED}(x) \wedge \mathsf{ED}(y) \wedge \mathsf{T}(t)$ (Endurant x is a part of y during t [3])
(**d**1) $CP(x,y) \triangleq \exists t\,(PRE(y,t)) \wedge \forall t\,(PRE(y,t) \rightarrow P(x,y,t))$ (Constant Part [3])
(12) $NEC(x,y) \rightarrow \mathsf{CSTR}(x) \wedge \mathsf{FD}(y) \wedge CP(x,y)$ (x is necessary in y)
(13) $OBG(x,y) \rightarrow \mathsf{CSTR}(x) \wedge \mathsf{FD}(y) \wedge CP(x,y)$ (x is obligatory in y)
(14) $\mathsf{CSTR}(x) \wedge \mathsf{FD}(y) \wedge P(x,y) \rightarrow NEC(x,y) \vee OBG(x,y)$

Classification Necessary constraints must always evaluate to true for an object to be considered to be classified by the concept with the constraint. In contrast, obligatory constraints may be violated while still considering the object to be classified by the concept. To handle this, we weaken the definition of classification (CF) from [4] and introduce two specific types of classification: artefactual classification, ACF, and proper classification, PCF.

Artefactual Classification embodies the idea that an artefact is intended to be of a certain type, or is created to conform to a particular specification or artefact species [6]. However, it may not always conform to it, for example, if it is broken down. *Proper* classification, on the other hand, assumes the notion of full conformance to the constraints of the description and applies to non-artefactual concepts, e.g., social concepts, and roles. Importantly, proper classification allows for a concept to be classified by another concept, allowing the construction of multi-level hierarchies describing concepts at different levels of abstraction.

(15) $CF(x,y,t) \rightarrow \mathsf{ED}(x) \wedge \mathsf{CN}(y) \wedge \mathsf{T}(t)$ (endurant x is classified by y at time t [4])
(16) $ACF(x,y,t) \rightarrow CF(x,y,t)$ (17) $PCF(x,y,t) \rightarrow CF(x,y,t)$
(18) $ACF(x,y,t) \rightarrow \mathsf{PAO}(x) \wedge \mathsf{AT}(y)$ (ACF holds between a physical artefactual object[4] and an artefact type)

Relationship Classification Moreover, we adopt the relationship classification relation from the second approach of [5], i.e. for a relationship with arity 2 there is a relation $CF(x,y,r,t)$ (extendable to different arities) meaning 'x and y are in the relationship r during t'. Since the formalisation of relationships is not the focus of this paper, we provide only a minimal definition. However, additional axioms could be introduced to formalise a particular theory, such as those of [5,8]. Since relationships are concepts, relationship classification implies the existence of an instance (RELI) that is classified by the relationship concept. The relation

the definition in [4] that a concept be defined by a single description nor does it impact the ability for different information objects to express a formal description in different ways.

[4]For simplicity we consider only physical artefacts in this paper; however, the formalisation presented can be easily extended to consider abstract artefacts such as Information Artefacts.

instance is *specifically constantly dependent* (*SD*) [3] on the objects in the relations (i.e. the relation can only be present while the related objects are present).

(19) $CF(x, y, r, t) \rightarrow \mathsf{ED}(x) \wedge \mathsf{ED}(y) \wedge \mathsf{REL}(r) \wedge \mathsf{T}(t)$

(20) $CF(x, y, r, t) \rightarrow \exists i\, \mathsf{RELI}(i) \wedge PCF(i, r, t) \wedge SD(i, x) \wedge SD(i, y)$

Constraint Evaluation Constraints and descriptions can be applied (i.e. evaluated for truth) in the context of an object. This is denoted by $\phi(x, t)$, which means 'the constraint/description ϕ is satisfied by x during t.' Therefore, the requirement that the constraints of concepts are satisfied by the objects classified by them can be considered as follows:

(21) $ACF(x, y, t) \wedge DF(y, \phi) \rightarrow \bigwedge_{\psi, NEC(\psi,\phi)} \psi(x, t)$

(All *necessary* constraints must be satisfied by an object that is artefactually classified by a concept)

(22)* $PCF(x, y, t) \wedge DF(y, \phi) \rightarrow \bigwedge_{\psi, P(\psi,\phi)} \psi(x, t)$

(*All* constraints must be satisfied by an object that is proper classified by a concept)

The problem with using (22) for proper classification is that it only works with strict classification, i.e., when each level of classification is fully described by the immediate level above. However, when working with multiple levels of classification, constraints may be defined that relate qualities, etc. across multiple levels and that cannot be satisfied at an intermediate level. This occurs with technical artefacts where the designs, e.g., specify particular requirements or nominal qualities, such as 'operating temperature', that the physical artefact is to fulfil. In traditional representations, the operating temperature would be defined as a necessary quality for the artefact kind 'Pump', while a subclass representing an artefact species of pump (i.e., a pump model) would constrain the range of the quality to enforce the requirement. This implies that the physical pump would cease to be a pump if its physical temperature strayed outside of the range of its design, or various other nonsensical outcomes depending on formalisation and commitments in use. This is due to conflating the qualities of the design (the requirements), with the qualities of the physical artefact.

A multi-level classification system separates these: there are distinct concepts for the artefact kind 'Pump', the artefact species (or pump models), and the concept of 'Pump Model'. The latter describes what all designs of pump—i.e., the artefact species which subclass 'Pump'—are to specify, such as the 'operating temperature' requirement distinct from the 'temperature' quality of all pumps. The two are naturally linked via a constraint, such as 'the temperature of a pump *should* remain within the operating temperature range defined by its pump model.'

To do so in a general fashion, the constraint between the requirement at one level and the quality at another must be definable. Such a constraint is not satisfied at the concept of 'Pump Model', nor at a specific model of pump, but rather at the level of the physical pump artefact. Therefore, proper classification cannot simply demand that *all* of the described constraints be satisfied by the classified object as it would prevent the definition of such multi-level constraints.

Artefactual classification does not exhibit this issue as it always classifies an artefact, never a concept, and thus always requires a complete description.

To support constraints across multi-level classification hierarchies, we replace (22) with (23) to take into account the nature of the constraints and whether or not

they can be evaluated in an object's context. Doing so maintains consistency when propagating constraints across classification levels via domain specific relationships by only evaluating constraints if the classified object is not a concept (hence fully described) or if the (abstract) quality or relationship has been previously introduced. The idea stems from the notion that the constraint that introduces the use of a quality or relationship is not the concept on which it should be applied, rather it should be applied at a lower level of abstraction. When applying the basic framework, where each level is fully described, the evaluation of all constraints will occur at the immediate level below as expected, as all the qualities and relationships will be introduced by the classifying concept.

(23) $PCF(x,y,t) \wedge DF(y,\phi) \rightarrow \bigwedge_{\psi,EV(\psi,\phi,x)} \psi(x,t)$

(24) $EV(\psi,\phi,x) \leftrightarrow P(\psi,\phi) \wedge \big(\neg \mathsf{CN}(x) \vee$
$(\mathsf{CN}(x) \wedge \forall q\,[US(q,\psi) \wedge q \in \mathcal{Q}_A \rightarrow \neg INTR(q,\phi)]) \vee$
$(\mathsf{CN}(x) \wedge \forall r\,[US(r,\psi) \wedge \mathsf{REL}(r) \rightarrow \neg INTR(r,\phi))]\big)$
(Constraint ψ of description ϕ can be evaluated in the context of object x)

Where *INTR* means 'introduced by' and *US* means 'used by', which are formalised later for the internal structure of descriptions and constraints.

Subconcepts A concept can be a subconcept of another, similar to subsumption between unary predicates [4]. Everything that is classified by a subconcept is classified by the parent concept. In particular, if an object is artefactually classified or proper classified by a concept, then it must also be artefactually classified or proper classified by any parent concepts. As a result, an object must satisfy the descriptions of all parent concepts (28, 29).

(25) $SB(x,y) \rightarrow \mathsf{CN}(x) \wedge \mathsf{CN}(y)$ (x is a subconcept of the concept y)

(**d2**) $SB(x,y) \triangleq \forall t\big[\exists z[CF(z,x,t)] \wedge \forall z[CF(z,x,t) \rightarrow CF(z,y,t)]\big]$ (adapted from [4])

(26) $SB(x,y) \wedge SB(y,z) \rightarrow SB(x,z)$ (Subconcepts are transitive)

(27) $\mathcal{P}(x,y,t) \wedge SB(y,z) \rightarrow \mathcal{P}(x,z,t)$, for $\mathcal{P} \in \{ACF, PCF\}$

(28) $ACF(x,y,t) \wedge SB(y,z) \wedge DF(z,\phi) \rightarrow \bigwedge_{\psi,NEC(\psi,\phi)} \psi(x,t)$ (from 21, 27)

(29) $PCF(x,y,t) \wedge SB(y,z) \wedge DF(z,\phi) \rightarrow \bigwedge_{\psi,EV(\psi,\phi)} \psi(x,t)$ (from 23, 27)

Constraint Structure Constraints must be defined in some constraint language, which we leave flexible in this framework. However, there are minimum requirements. Since the descriptions (and, hence, constraints) must be anchored in the ontology [4], the language must support the ability to: (a) reference the primitives of the ontology; (b) reference well known concept names and object names from the domain of quantification (treated as constants); (c) reference a special constant, e.g. SELF, representing the contextual object with respect to which the constraints are evaluated; (d) refer to qualities and values (regions of a quality space); (e) include logical operators and quantifiers, such as conjunction, disjunction, universal quantification, existential quantification, etc.

These requirements allow defining key concepts and constraints: (a) relationships linked to primitive predicates, e.g., domain specific part-of relationship associated with the primitive parthood predicate (P); (b) constraints on the existence of a quality on the classified objects, e.g., wheels require a diameter; (c) constraints on the values of a quality, e.g., 10 inch wheels have a diameter within the 10

inch region of the quality space. This may be inexact to allow for tolerances, e.g., $10 \pm 1/4$ inches (i.e., nominal qualities); (d) constraints on the existence of entities of certain types, e.g., a concept may require a specific relationship to an object of another type, while relationships may have constraints on the existence of mediating entities; (e) constraints based on temporal aspects and perdurants such as process instances, e.g., some constraints on a piece of equipment may only apply while the equipment is participating in a "running" process.

To keep the theory agnostic of any particular constraint language, we re-purpose the 'used by' relation (US) of [4] to characterise the contents of constraints in terms of the qualities, relationships, and concepts that they reference. The re-purposed relation is formalised almost identically to the original, allowing the original to be derived from the contents of the description, rather than as primitive on descriptions themselves. In particular, the axiom that enforces the use of a concept in its own description (33) remains applicable as it embodies the idea that the constraints are in the context of an object of that concept. Moreover, each constraint uses the defined concept (34) since the reference to the special constant SELF implicitly uses the concept of the description and is the starting point of each constraint. In addition, we specify the 'introduced by' relation ($INTR$) to identify constraints that introduce qualities or relations, not just use them. This allows for the identification of the point in a classification hierarchy that a quality or relationship is introduced and not just referenced. When dealing with a specific constraint language, this relation should be defined in terms of that language.

(30) $US(x,y) \rightarrow (x \in \mathcal{Q} \vee \mathsf{CN}(x)) \wedge (\mathsf{DS}(y) \vee \mathsf{CSTR}(y))$ (The quality or concept[5] x is used by the description or constraint y)

(31) $INTR(x,y) \rightarrow (x \in \mathcal{Q} \vee \mathsf{REL}(x)) \wedge (\mathsf{FD}(y) \vee \mathsf{CSTR}(y))$ (The quality or relationship x is introduced by y)

(32) $INTR(x,y) \rightarrow US(x,y)$ (An introduced quality/relationship is used by the constraint)

(33) $DF(x,y) \rightarrow US(x,y)$ (A concept is used by the description that defines it [4])

(34) $DF(x,y) \wedge P(c,y) \rightarrow US(x,c)$ (All parts/constraints of a description use the concept the description defines)

(35) $US(x,y) \rightarrow (PRE(y,t) \rightarrow PRE(x,t))$ (Used concepts must be present when the using entity is present [4])

(36) $US(x,y) \wedge \mathsf{FD}(y) \leftrightarrow \exists c\, \mathsf{CSTR}(c) \wedge P(c,y) \wedge \mathsf{FD}(y) \wedge US(x,c)$

(37) $INTR(x,y) \wedge \mathsf{FD}(y) \leftrightarrow \exists c\, \mathsf{CSTR}(c) \wedge P(c,y) \wedge INTR(x,c)$

A special handling of quality types is required as they are universals. Here we use a treatment similar to DOLCE [3] which considers a second order axiom as syntactic sugar for a finite list of first-order axioms. This can be achieved since, for a given ontology, we can assume a finite set of quality types, $\mathcal{Q}$. In addition, we can posit sets representing the partitioning of quality types as physical, abstract, or temporal as $\mathcal{Q}_P$, $\mathcal{Q}_A$, and $\mathcal{Q}_T$, respectively. However, this approach mixes constraints over universals with elements in the domain of quantification and requires modification of the universals of the ontology to incorporate domain specific quality types. An alternative would integrate quality types into the domain

[5]Since relationships are concepts they do not have to be explicitly mentioned.

of quantification in a similar fashion to concepts and relationships; this will be performed in future work.

Example Constraints Consider the simplified representation of a type of pump (such as a particular model of centrifugal pump) and its specification. It may include constraints on necessary parts[6], for example, an impeller of a certain type. The impeller itself may define necessary constraints such as its diameter being of a certain dimension (within some tolerance). Moreover, the pump may include constraints on its operating temperature and flow rate[7] such that they must stay within certain ranges but can be violated. For example, if the connecting pipe is broken the flow rate will drop without affecting the object's status as a pump. The referenced quality types (e.g., operating-temp) are assumed to have been incorporated into the ontology. Example 1 illustrates these constraints using a first-order logic-like language where terms in SMALL CAPS represent names of concepts and qualities that are referenced in the constraint. As the constraints are reified in the domain of quantification, they are illustrated as symbols with their content defined in the boxes. In addition, the following shorthand is used for classification: $\text{CN}(x,t) \rightarrow CF(x,\text{CN},t)$, $\text{R}(x,y,t) \rightarrow CF(x,y,\text{R},t)$ The relations *qt* and *ql* from DOLCE [3] indicate that something is a quality and a quale (i.e., a value of a quality), respectively. For brevity, the quantification of time inside constraint expressions is in the context of the object for which the constraint is evaluated. When no explicit quantification is specified it is implicitly: $\forall t\, PRE(\text{SELF},t)$. For example, in ψ_{PUMP1}, a pump must have an impeller only for those times at which the pump is present.

Example 1 (Definition of a pump concept and associated constraints)

$$\mathsf{AS}(\text{PUMP}) \wedge DF(\text{PUMP},\phi_{\text{PUMP}}) \wedge NEC(\psi_{\text{PUMP1}},\phi_{\text{PUMP}}) \wedge OBG(\psi_{\text{PUMP2}},\phi_{\text{PUMP}}) \wedge OBG(\psi_{\text{PUMP3}},\phi_{\text{PUMP}})$$

$$\psi_{\text{PUMP1}} = \boxed{\exists x\, \text{HAS-PART}(\text{SELF},x,t) \wedge \text{IMPELLER}(x,t)}$$

$$\psi_{\text{PUMP2}} = \boxed{\exists q\, qt(q,\text{SELF}) \wedge \mathsf{operating\text{-}temp}(q) \wedge ql(v,q,t) \wedge P(v,\underline{100°\text{-}200°\text{C}},t)}$$

$$\psi_{\text{PUMP3}} = \boxed{\exists q\, qt(q,\text{SELF}) \wedge \mathsf{flow\text{-}rate}(q) \wedge ql(v,q,t) \wedge P(v,\underline{10 \pm 0.5\ \text{L/m}},t)}$$

$$\mathsf{AS}(\text{IMPELLER}) \wedge DF(\text{IMPELLER},\phi_{\text{IMPELLER}}) \wedge NEC(\psi_{\text{IMPELLER1}},\phi_{\text{IMPELLER}})$$

$$\psi_{\text{IMPELLER1}} = \boxed{\exists q\, qt(q,\text{SELF}) \wedge \mathsf{diameter}(q) \wedge ql(v,q,t) \wedge P(v,\underline{10 \pm 1/4"},t)}$$

$$\mathsf{REL}(\text{HAS-PART}) \wedge DF(\text{HAS-PART},\phi_{\text{HAS-PART}}) \wedge NEC(\psi_{\text{HAS-PART1}},\phi_{\text{HAS-PART}})$$

$$\psi_{\text{HAS-PART1}} = \boxed{\text{SELF}(x,y,t) \rightarrow P(y,x,t)}$$

We now briefly introduce the SLICER framework before defining how it can be integrated to improve the granularity of the descriptions.

[6] For simplicity of the example we ignore that parts of an artefact can be replaced.

[7] Such constraints would only apply while the pump is running; however, since we do not cover (artefactual) processes in the present work for brevity, such distinctions are not made.

3. Using the SLICER Relationship Framework for Relationship Concepts

The SLICER (Specification with Levels based on Instantiation, Categorisation, Extension, and Refinement) framework was developed in the context of complex domain models of the engineering life cycle to reduce the modelling load via flexible meta-level modelling techniques.

A hierarchy of layers is used to separate ontological from representational ("linguistic") aspects, a concept known as multilevel modelling. In the engineering life cycle, a higher level generally expresses the relationship between an entity and its definition (or description), correlating with the intentional design process [9].

A *level of description* can be established either by instantiation, or by enriching the vocabulary used to formulate the descriptions. This corresponds to the concept of *extension* in specialisation hierarchies [10]: if a subclass describes additional properties (attributes, relationships, etc.) then these properties can be used to impose constraints on its specification and behaviour.

To support the purpose of describing joint metamodels in interoperability scenarios, SLICER is based on a flexible notion of levels identified by applying the semantic relationships below.

Instantiation and Specialisation Levels of description are dynamically derived based on finer distinctions of the instantiation and specialisation relations. Specialisation relationships *extend* the original class (by adding attributes, associations, or behaviour, i.e., constraints or state change differentiation, $SpecX$) or *refines* it (by adding granularity to the description, $SpecR$). Of the two, only $SpecX$ introduces a new model level.

Similarly, instantiation is characterised as *Instantiation with Extension* ($InstX$, allowing additional properties, etc.) or *Normal Instantiation* ($InstN$). Instantiation always introduces additional model levels. Objects created through $InstN$ cannot be instantiated further and form the most basic-level of individuals in a model.

Categories are concepts providing external ("secondary") grouping of entities based on common properties and/or explicit enumeration of members. We do not discuss them further here.

Specifications are expressed via the *Subset by Specification* (SbS) relation. The specification class (for example EquipmentModel) exists at the same level as the type it refers to as it can define constraints with respect to that type. Specifications and Categories relate to two common ways in which powertypes are applied [2].

Descriptions A description, e.g. a set of constraints, can refer only to the properties specific to its object (or, if the description is for a specification, the properties of the type associated with the specification) and are inherited through specialisation, while instances of a type must satisfy its description.

This framework can now be used on top of the basic formalisation from Section 2. That formalisation allowed for the definition of concepts, their (intended) instances, and the checking of constraints between an entity and its classifying concepts. As a result, the definitions of SLICER relationships can be reformulated within the ontological framework, by defining the relationships as instances of REL, linking them to appropriate primitive relations, and incorporating the SLICER

axioms as constraints within their descriptions. This way different information representations, including information models and data models at different levels of expressivity, can be formalised within the same ontological framework.

The following definitions validate the ontological framework against the requirements of specifying SLICER relationships.

3.1. General Instantiation and Specialisation Relationships of SLICER

The formulation of the core relationships of general instantiation (*Inst*) and specialisation (*Spec*) relationships is shown in Eqs. (38) and (39), respectively. Below, *Inst* is a relationship concept (REL) defined by (*DF*) the description ϕ_{INST} with the necessary constraint (*NEC*) labelled ψ_{INST1}. ψ_{INST1} is a reified expression indicating that instances of the relationship imply adherence to the standard classification relation, cf. (15), tying the domain relationship to the ontological primitives. The general specialisation relationship is defined analogously.

$$\begin{aligned} &\text{REL}(\text{INST}) \wedge DF(\text{INST}, \phi_{\text{INST}}) \wedge NEC(\psi_{\text{INST1}}, \phi_{\text{INST}}), \\ &\psi_{\text{INST1}} = \boxed{\text{SELF}(x, y, t) \rightarrow CF(x, y, t)} \end{aligned} \tag{38}$$

$$\begin{aligned} &\text{REL}(\text{SPEC}) \wedge DF(\text{SPEC}, \phi_{\text{SPEC}}) \wedge NEC(\psi_{\text{SPEC1}}, \phi_{\text{SPEC}}), \\ &\psi_{\text{SPEC1}} = \boxed{\text{SELF}(x, y, t) \rightarrow SB(x, y, t)} \end{aligned} \tag{39}$$

3.2. Relationships Incorporating Extension and Refinement

SLICER introduces more specific relationships based on whether an object *extends* (adding additional attributes, behaviour, etc.), *refines* (adding granularity, e.g. by restricting the range of an attribute), and/or *instantiates* (i.e. assigns values to its attributes) another [2]. These relationships can be defined as shown in Eqs. (40) and (41). All specialisations may include refinement, while *SpecX* must incorporate additional attributes, i.e. qualities or relationships in this context (refer Eq. (41)). To support this we add a constraint that enforces the propagation of constraints across all *Spec* relationships such that necessary constraints remain necessary and obligations remain obligations (Eq. (40)). This remains consistent with the evaluation of descriptions of parent types during classification. Although it would be symmetrical with *SpecX* to enforce refinement in *SpecR*, doing so would disallow the ability to extend the vocabulary of concepts in the case where no refinement to the modelled attributes is included. For the same reason, we have not defined the identity of concepts based on their intension, that is, the constraints included in their descriptions. (However, a specific application could choose to include additional axioms to enforce such a constraint.)

$$\begin{aligned} &DF(\text{SPEC}, \phi_{\text{SPEC}}) \wedge NEC(\psi_{\text{SPEC2}}, \phi_{\text{SPEC}}), \\ &\psi_{\text{SPEC2}} = \boxed{\begin{array}{c} \text{SELF}(x, y, t) \wedge DF(x, \phi_x) \wedge DF(y, \phi_y) \rightarrow \\ \forall c \left[\mathcal{P}(c, \phi_y) \rightarrow \mathcal{P}(c, \phi_x)\right] \end{array}} \text{ for } \mathcal{P} \in \{NEC, OBG\} \end{aligned} \tag{40}$$

$$\begin{aligned}
&\mathsf{REL}(\textsc{SpecX}) \wedge DF(\textsc{SpecX}, \phi_{\textsc{SpecX}}) \wedge NEC(\psi_{\textsc{SpecX1}}, \phi_{\textsc{SpecX}}) \\
&\qquad \wedge NEC(\psi_{\textsc{SpecX2}}, \phi_{\textsc{SpecX}}), \\
&\psi_{\textsc{SpecX1}} = \boxed{\textsc{self}(x,y,t) \rightarrow \textsc{Spec}(x,y,t)}, \\
&\psi_{\textsc{SpecX2}} = \boxed{\begin{aligned}&\textsc{self}(x,y,t) \wedge DF(x,\phi_x) \wedge DF(y,\phi_y) \rightarrow \\ &\qquad \exists z\,(\mathsf{REL}(z) \vee z \in \mathcal{Q}) \wedge US(z,\phi_x) \wedge \neg US(z,\phi_y)\end{aligned}}
\end{aligned} \tag{41}$$

A similar characterisation can be given of *InstX* and *InstN*, as illustrated in Eqs. (42) and (43). The intended application of SLICER is for the definition of artefacts, so we can define *InstN* in terms of artefactual classification.[8]

$$\begin{aligned}
&\mathsf{REL}(\textsc{InstX}) \wedge DF(\textsc{InstX}, \phi_{\textsc{InstX}}) \wedge NEC(\psi_{\textsc{InstX1}}, \phi_{\textsc{InstX}}) \\
&\qquad \wedge NEC(\psi_{\textsc{InstX2}}, \phi_{\textsc{InstX}}), \\
&\psi_{\textsc{InstX1}} = \boxed{\textsc{self}(x,y,t) \rightarrow \textsc{Inst}(x,y,t) \wedge PCF(x,y,t)}, \\
&\psi_{\textsc{InstX2}} = \boxed{\begin{aligned}&\textsc{self}(x,y,t) \wedge DF(x,\phi_x) \wedge DF(y,\phi_y) \rightarrow \\ &\qquad \exists z\,(\mathsf{REL}(z) \vee z \in \mathcal{Q}) \wedge US(z,\phi_x) \wedge \neg US(z,\phi_y)\end{aligned}}
\end{aligned} \tag{42}$$

$$\begin{aligned}
&\mathsf{REL}(\textsc{InstN}) \wedge DF(\textsc{InstN}, \phi_{\textsc{InstN}}) \wedge NEC(\psi_{\textsc{InstN1}}, \phi_{\textsc{InstN}}) \\
&\psi_{\textsc{InstN1}} = \boxed{\textsc{self}(x,y,t) \rightarrow \textsc{Inst}(x,y,t) \wedge ACF(x,y,t)}
\end{aligned} \tag{43}$$

This characterisation implies that only concepts can be in an *InstX* relationship with another concept, while non-concepts are the instances in *InstN* relationships. Since non-concepts do not have descriptions, they cannot be extended with constraints on additional quality types or relationships. Since the ontological framework ensures that qualities have values (or quales), the assignment of values to qualities does not need to be included in the constraints of *InstN*. Similarly, given the axioms adopted for relations above, the existence of a value for a relationship required by a constraint is already enforced; therefore, it is not necessary to include a constraint for this in the description of *InstN*.

3.3. Subset by Specification

Another important relation in SLICER is Subset by Specification (*SbS*), which is a form of powertyping relation that states the instances of the specification type are subconcepts of the associated concept [2]. Such a construct is frequently encountered in design and manufacturing settings. A minimal form of this relationship can be defined within the ontological framework as shown in Eq. (44).

[8] This may not be the case for the full framework including roles and functions.

$$\begin{aligned}
&\mathsf{REL}(\textsc{SbS}) \wedge DF(\textsc{SbS}, \phi_{\textsc{SbS}}) \wedge NEC(\psi_{\textsc{SbS1}}, \phi_{\textsc{SbS}}) \wedge NEC(\psi_{\textsc{SbS2}}, \phi_{\textsc{SbS}}),\\
&\psi_{\textsc{SbS1}} = \boxed{\textsc{self}(x, y, t) \rightarrow \mathsf{CN}(x) \wedge \mathsf{CN}(y)},\\
&\psi_{\textsc{SbS2}} = \boxed{\textsc{self}(c, c', t) \wedge \mathit{Inst}(x, c, t) \rightarrow \textsc{Spec}(x, c', t)}
\end{aligned} \tag{44}$$

3.4. Constraint Propagation Across Instantiation Relationships

Finally, SLICER introduces an intuitive method of propagating constraints across multiple levels of instantiation/classification to where they can be evaluated. The propagation is determined by whether a constraint can be evaluated for the current object based on the presence or absence of appropriate attributes [2]. Within the ontological framework we can make a similar distinction based on the types of qualities and relations used by a constraint along with the *INTR* relation.

For example, a higher level concept may define constraints based on physical qualities; however, concepts are non-physical and cannot have physical qualities in DOLCE [3]. Therefore, when a concept instantiates the higher level concept, the constraints using physical qualities are propagated to the instantiating concept. Then, when a physical object instantiates the bottom-most concept, the constraints can be evaluated in the context of a physical object. A similar situation occurs for abstract and temporal qualities, except that other non-physical objects (not just concepts) can have abstract qualities. Therefore, the propagation only occurs for the abstract qualities that have not yet been introduced. Eq. (45) illustrates how the propagation can be defined on the descriptions of the instantiation relationships using a shorthand to abstract over necessary and obligatory constraints.

$$\begin{aligned}
&DF(\textsc{Inst}, \phi_{\textsc{Inst}}) \wedge NEC(\psi_{\textsc{Inst2}}, \phi_{\textsc{Inst}}),\\
&\psi_{\textsc{Inst2}} = \boxed{\begin{aligned}
&\textsc{self}(x, y, t) \wedge DF(x, \phi_x) \wedge DF(y, \phi_y) \rightarrow\\
&\quad \forall c, q\,[\mathcal{P}(c, \phi_y) \wedge US(q, c) \wedge q \in \mathcal{Q}_P \cup \mathcal{Q}_T \rightarrow \mathcal{P}(c, \phi_x)] \wedge\\
&\quad \forall c, q\,[\mathcal{P}(c, \phi_y) \wedge US(q, c) \wedge \neg \mathit{INTR}(q, \phi_y) \wedge q \in \mathcal{Q}_A \rightarrow \mathcal{P}(c, \phi_x)] \wedge\\
&\quad \forall c, r\,[\mathcal{P}(c, \phi_y) \wedge US(r, c) \wedge \neg \mathit{INTR}(r, \phi_y) \wedge \mathsf{REL}(r) \rightarrow \mathcal{P}(c, \phi_x)]
\end{aligned}}\\
&\text{for } \mathcal{P} \in \{\mathit{NEC}, \mathit{OBG}\}
\end{aligned} \tag{45}$$

Constraint Propagation Example Consider extending the constraints of Example 1 to utilise the SLICER model. Here the operating temperature constraint is not defined solely as operating temperature; instead, it is defined as a comparison between the values of two qualities that are introduced in different concepts such that the constraint is propagated over two instantiation relationships. The constraint is defined between the operating temperature (physical) quality, defined on PumpModel, and the (actual) temperature quality, defined on Pump (the superclass of all pump types). The operating temperature is a *design* (or *nominal*) quality that can be constrained to a specific region by the different pump models/types. The concept definitions and their constraints are illustrated in Example 2.

Example 2 (Definition of pump concept and constraints using SLICER)

$$\begin{aligned}
&\mathsf{AK}(\textsc{Pump}) \wedge DF(\textsc{Pump}, \phi_{\textsc{Pump}}) \wedge NEC(\psi_{\textsc{Pump1}}, \phi_{\textsc{Pump}})\\
&\psi_{\textsc{Pump1}} = \boxed{\exists x\; qt(x, \textsc{self}) \wedge \mathsf{temperature}(x)}
\end{aligned}$$

$$\begin{aligned}
&\mathsf{CN}(\textsc{PumpModel}) \wedge \textsc{SbS}(\textsc{PumpModel}, \textsc{Pump}, t) \wedge \mathit{DF}(\textsc{PumpModel}, \phi_{\textsc{PumpModel}}) \\
&\qquad \wedge \mathit{NEC}(\psi_{\textsc{PumpModel1}}, \phi_{\textsc{PumpModel}}) \\
&\qquad \wedge \mathit{OBG}(\psi_{\textsc{PumpModel2}}, \phi_{\textsc{PumpModel}}) \\
&\psi_{\textsc{PumpModel1}} = \boxed{\exists q\, qt(q, \textsc{self}) \wedge \mathsf{operating\text{-}temp}(q)} \\
&\psi_{\textsc{PumpModel2}} = \boxed{\begin{array}{l} qt(q_{ot}, \textsc{self}) \wedge \mathsf{operating\text{-}temp}(q_{ot}) \wedge ql(v_{ot}, q_{ot}, t) \wedge \\ \quad qt(q_t, \textsc{self}) \wedge \mathsf{temperature}(q_t) \wedge ql(v_t, q_t, t) \rightarrow P(v_t, v_{ot}, t) \end{array}} \\
&\mathsf{AS}(\text{C12Pump}) \wedge \mathit{SpecX}(\text{C12Pump}, \textsc{Pump}, t) \wedge \mathit{InstX}(\text{C12Pump}, \textsc{PumpModel}, t) \\
&\qquad \wedge \mathit{DF}(\text{C12Pump}, \phi_{\text{C12Pump}}) \wedge \mathit{NEC}(\psi_{\text{C12Pump1}}, \phi_{\text{C12Pump}}) \\
&\qquad \wedge \mathit{NEC}(\psi_{\text{C12Pump2}}, \phi_{\text{C12Pump}}) \\
&\psi_{\text{C12Pump1}} = \boxed{\exists x\, \textsc{has-part}(\textsc{self}, x, t) \wedge \textsc{impeller}(x, t)} \\
&\psi_{\text{C12Pump2}} = \boxed{qt(q, \textsc{self}) \wedge \mathsf{operating\text{-}temp}(q) \rightarrow ql(\underline{100°\text{-}200°\text{C}}, q, t)}
\end{aligned}$$

Therefore, to be a proper instance of C12Pump, a physical pump's temperature must be consistent with the operating temperature range of 100°-200°C. If the obligation is not fulfilled, the pump is still considered to be a C12Pump due to the artefactual classification; however, it does not conform to its specification.

The constraints defined on PumpModel will be propagated to the description of C12Pump via the instantiation relationship, thereby making it an obligatory constraint for all pumps. This demonstrates the ability to tailor the semantics in the ontology to specific domains entirely within the domain of discourse. Different domains may use different sets of semantics side-by-side: indeed, as defined here, the SLICER semantics are specifically suited to artefacts.

4. Related Work

Wang et al. [11] have conducted ontological analysis of software systems based on a requirements engineering perspective. They defined multiple levels of detail, bottoming out in code, where each level is *constitutedBy* lower-level pieces of software in a relationship defined by Baker [12]. Moreover, each level (excluding code) is associated with a specification or description. To do so they define a relationship *intendedToImplement* that corresponds roughly to our *ACF* relation; the "intended" implying that the implementation may not be exactly correct. Relations *intendedToSatisfy* and *presupposes* are used to relate different levels of requirements and separate out environment assumptions. However, the specifications are not broken down further and no axiomatisation is given.

Guarino and Melone [13] informally discuss the ontological status of design objects, based on the viewpoint of architects, and therefore assuming that design objects represent physical artifacts. The basic role is that of *design element* which, installed in a particular position, serves as a *design component*. A *conventional system component*, as in [6] is a particular location considered by the designer and

a *physical system component* is an actual physical object resulting from the design. While Guarino and Melone identify these different design objects, they do not consider the definition or content of the specifications.

The work by Sanfilippo et al. [14] is most similar to our own, focusing on a structural decomposition of design objects that bottoms out in quality spaces. The axiomatisation of (artefactual) concepts is based on some of the same basis as our work such as DOLCE [3] and previous work on roles [4]. The characterisation differentiates and compares design concepts vs. requirements concepts, but does not consider the relationship to instances of the concepts. In contrast, we differentiate necessary and obligatory constraints in the context of how they are fulfilled by artefact instances. In addition, our characterisation of concepts goes further than a simple group of properties as we explicitly associate the concepts with constraints.

More recently, Sanfilippo et al. [9] investigated the foundational ontological basis of nominal and actual qualities through three possible representations: Nominal qualities as qualities, Nominal qualities as properties, and Nominal qualities as descriptions. Nominal qualities are highly related to engineering design as they define constraints to which an artefact, with actual qualities, is expected to conform to some degree. Our approach fits into the latter representation, in which nominal qualities are defined by descriptions, where we characterise the content of the descriptions as formal, executable constraints. While we have only illustrated simple constraints, including what would be considered nominal qualities, our approach is capable of (and intended for) much more complex constraints.

The work of [15,16] does not formalise the description of an artefact, but merely its interface for assembly of cosimulation processes, but could serve as another use case addressing one specific phase of the artefact life cycle. While their GOPPRR approach supports the incorporation of different domains or information representations into a single model, there is limited to no ontological commitment.

In [17], a group of modular ontologies built on the Basic Formal Ontology upper ontology are described that cover the engineering life cycle, including 'design specifications', etc. The approach treats design specifications as Information Content Entities (in the terms of the BFO-conforming Information Artifact Ontology), rather than characterising their content as generic constraints that can be applied to their instances. Also, as pointed out in [9], the BFO-based approach may have difficulty representing nominal qualities (and, hence, more complex constraints) as the Information Content Entities must be defined for things that may not yet exist, which is in conflict with their definition.

5. Conclusion

In this work, we have brought together two threads of our work on large-scale model-driven interoperability: the framework for artefacts and roles, based on an extension of the DOLCE ontology [1], and the SLICER (Specification with Levels based on Instantiation, Categorisation, Extension, and Refinement) approach providing the semantic building blocks for the cleanly structured representation of industrial artefact domains in detail [2]. A key innovation of SLICER is the explicit handling of artefact *descriptions*, and based on Masolo and others' work on

social roles [4] we have provided a framework for the formalisation of descriptions that enables the modelling and management of domain specific constraints on technical artefacts. We have shown how to represent necessary and obligatory constraints in the context of evolving information, and obtained a framework capable of validating information models across multiple life cycle stages w.r.t. specific domain requirements. The framework is being incorporated in our F-Logic based transformation and validation environment [2]. Future work will use this to perform tasks such as requirements verification, incorporate the handling of functions and roles, and study the semantic SLICER relationship types as meta-ontological properties in the plant life cycle domain. Also, adopting a reduced ontological commitment following the (Constructive) Descriptions and Situations framework [18] would allow this work to be applied across different foundational ontologies in support of semantic interoperability for complex constraints.

References

[1] Jordan A, Selway M, Mayer W, Grossmann G, Stumptner M. An Ontological Core for Conformance Checking in the Engineering Life-cycle. In: Proc. FOIS 2014; 2014. .
[2] Selway M, Stumptner M, Mayer W, Jordan A, Grossmann G, Schrefl M. A conceptual framework for large-scale ecosystem interoperability and industrial product lifecycles. Data Knowl Eng. 2017;109:85-111.
[3] Masolo C, Borgo S, Gangemi A, Guarino N, Oltramari A, Schneider L. WonderWeb Deliverable D17: The WonderWeb Library of Foundational Ontologies. In: TR-NRC, Institute of Cognitive Sciences and Technology. Italy; 2003. .
[4] Masolo C, Vieu L, Bottazzi E, Catenacci C, Ferrario R, Gangemi A, et al. Social Roles and their Descriptions. In: Proc. KR '04; 2004. p. 267-77.
[5] Masolo C, Guizzardi G, Vieu L, Bottazzi E, Ferrario R. Relational roles and qua-individuals. In: Proc. AAAI Fall Symposium on Roles; 2005. p. 103-12.
[6] Guarino N. Artefactual Systems, Missing Components and Replaceability. In: Franssen M, et al., editors. Artefact Kinds. Springer; 2014. p. 191-206.
[7] Mizoguchi R, Kitamura Y, Borgo S. Towards a Unified Definition of Function. In: Proc. FOIS; 2012. p. 103-16.
[8] Guarino N, Guizzardi G. "We Need to Discuss the Relationship": Revisiting Relationships as Modeling Constructs. In: Proc. CAiSE; 2015. p. 279-94.
[9] Sanfilippo EM, et al. A Foundational View on Nominal and Actual Qualities in Engineering. In: Proc. FOIS. Cape Town; 2018. .
[10] Schrefl M, Stumptner M. Behavior Consistent Specialization of Object Life Cycles. ACM TOSEM. 2002;11(1):92-148.
[11] Wang X, Guarino N, Guizzardi G, Mylopoulos J. Towards an Ontology of Software. In: Proc. FOIS 2014. Rio de Janeiro; 2014. .
[12] Baker LR. The Ontology of Artifacts. Philos Explor. 2004;7:99-112.
[13] Guarino N, Melone MRS. On the ontological status of design objects. In: 1st Workshop on AI and Design at AIIA; 2015. .
[14] Sanfilippo E, Masolo C, Porello D. Design knowledge representation: an ontological perspective. In: 1st Workshop on AI and Design at AIIA; 2015. .
[15] Wang H, Wang G, Lu J, Ma C. Ontology Supporting Model-Based Systems Engineering Based on a GOPPRR Approach. In: Proc. WorldCIST'19. Cham: Springer; 2019. p. 426-36.
[16] Lu J, Wang G, Törngren M. Design Ontology in a Case Study for Cosimulation in a Model-Based Systems Engineering Tool-Chain. IEEE Systems Journal. 2020;14(1):1297-308.
[17] Otte JN, et al. An ontological approach to representing the product life cycle. Applied Ontology. 2019;14(2):179-97. Available from: https://doi.org/10.3233/AO-190210.
[18] Gangemi A. Norms and plans as unification criteria for social collectives. Auton Agents Multi Agent Syst. 2008;17(1):70-112.