A core reference ontology for the customer relationship domain

Applied Ontology 0 (2012) 1–0
IOS Press

A Core Reference Ontology for the Customer Relationship Domain *

Diego Magro and Anna Goy **
Università di Torino, Dipartimento di Informatica, Corso Svizzera 185, 10149, Torino, ITALY
E-mail: {magro, goy}@di.unito.it

Abstract.
Customer Relationship Management (CRM) has emerged as an important strategy that companies should implement in order to build profitable and stable relationships with their customer. The domain of CRM has peculiar chracteristics: a CRM strategy is largely independent from the specific market sector, it requires multiple units cooperation, it implies the management of a huge amount of knowledge, it is fruitfully supported by software solutions, and finally it implies the integration of human and machine activities. These characteristics suggest that both companies aiming at implementing an efficient CRM strategy, and software houses offering ICT solutions supporting CRM would take great advantage from a common semantic model of CRM. The main contribution of this paper is thus the proposal of O-CREAM-v2, a core reference ontology for the CRM domain, specifically targeted to Small and Medium Sized Enterprises (SME). The design of O-CREAM-v2 has been based on requirements mainly elicited from a domain analysis, which considered the way the involved actors talk about CRM within their business, by analyzing documents and interviews with representatives of SME and ICT companies. Moreover, in order to guarantee accuracy in the definition of the basic concepts and to support interoperability within/between companies, O-CREAM-v2 has been developed within the framework provided by the well-known DOLCE foundational ontology, together with three DOLCE extensions, i.e.the ontology of Descriptions and Situations, the Ontology of Information Objects, and the Ontology of Plans. O-CREAM-v2 is composed by two layers: an *upper core*, which models more general concepts and relations, and can be useful also in business domains other than CRM, and a *lower core*, representing concepts and relations specific to the CRM domain. The content requirements defined by the domain analysis pointed out that an ontology for the CRM domain has to account for both particulars (such as activities, offers, sales, etc.) and information about them (customer records, reports about sales, an so on). Moreover, since CRM is typically supported by software tools, O-CREAM-v2 includes the formal characterization of software applications. Thus, O-CREAM-v2 is structured into five modules: Relationships, Knowledge, Activities (all three spanning both the upper and the lower core), Software and Miscellaneous (both limited to the upper core). The five O-CREAM modules are described in details in the paper. The discussion is concluded by mentioning two possible exploitation perspectives for O-CREAM-v2, which could be the basis for building: (a) Web-based repositories supporting the mediation between supply and demand of CRM-related tools; (b) tools supporting users in building the formal representations of resources in ontology-based IR systems and in the semantic search engines application field.

Keywords: Customer Relationship Management, CRM, Customer Relationships, Business Knowledge, Business Activities, Software

1. Introduction

Customer Relationship Management (CRM) Freeland [2005] has emerged in the last decades as an important aspect that all companies should take into account in order to understand and anticipate customer needs, in the perspective of acquiring new customers and building profitable and stable relationships with "old" ones by increasing their fidelity. In particular, a key feature of the CRM philosophy is the so called one-to-one marketing perspective, i.e. taking into consideration the single customer and establishing personalized relationships with her, by producing personalized offers, pricing and payment policies, as well as delivery modalities, packaging, after-sale services, etc.

In order to achieve the above mentioned goals, a huge amount of knowledge about customers is needed, coupled with the capability of analyzing such data and exploiting them in a CRM perspective. Moreover,

*PREPRINT VERSION. Cite as: Diego Magro and Anna Goy, A Core Reference Ontology for the Customer Relationship Domain, Applied Ontology 7(2012), pp. 1-48, IOS Press, DOI 10.3233/AO-2012-0102
**Corresponding Author: Diego Magro, Dipartimento di Informatica, Corso Svizzera 185, 10149, Torino, ITALY. E-mail: magro@di.unito.it

in order to carry on communications activities, such as contacts with customers, organization of the sale force, targeted marketing campaigns, and so on - that are of major importance within CRM - a wide range of heterogeneous technologies and services are needed, ranging from more classical communication media (e.g., phone, fax) to e-mail and Web applications. In particular, in order to actually implement an effective CRM strategy, taking into account all mentioned aspects, different integrated enabling technologies are required, including Databases Management Systems and Data Warehouses, ERP systems, Business Intellingence tools, and Web-based applications.

Another interesting aspect of the CRM approach is that it implies a cross-department view, since different business units should cooperate in supporting CRM: Marketing, Sales, Logistics and Finance, Customer Service, etc. Moreover, CRM is largely independent from the specific market sector the company operates in: CRM has become crucial for cars resellers, as well as for typical food producers, for security devices installers, as well as for service providers.

Finally, within CRM, human activities are typically coupled with machine supported processes; e.g. the configuration of personalized offers, just to mention a very simple case, is typically supported by a software system, but requires human intervention.

In summary, CRM is an emerging area, requires multiple units cooperation, is independent from the specific market sector, requires the acquisition and elaboration of a huge amount of knowledge, is fruitfully supported by software solutions, and implies the integration of human and machine activities: these characteristics suggest that all the involved actors - e.g. companies aiming at implementing an efficient CRM strategy as well as software houses offering ICT solutions supporting CRM - would greatly benefit from a common language, a shared set of concepts, and a common model of CRM.

In particular, if CRM were supported by semantic technologies, it would be possible to reason on the knowledge about customers, and this would offer a much more effective support to decision making, to marketing-related strategies, and to other customer-oriented enterprise policies.

Another important aspect to take into account when implementing a CRM strategy is the choice of the ICT tools that support it. In particular, finding the most suited CRM software solution can be a challenging task, especially for small sized enterprises aiming at adopting a CRM approach. A formally defined shared set of concepts could be the basis for a mediation service between supply and demand of software tools supporting CRM.

The CRM domain is also characterized by a great heterogeneity in the nature of the involved concepts, which range from activities, to software tools, from business knowledge to customer relationships (see Section 2). This heterogeneity requires a unifying framework able to provide the basis for modeling CRM-related concepts (and their relations) in a coherent and reusable way.

In the literature, there are a lot of efforts aimed at modeling concepts related to enterprise activities and business (see, for instance, Rittgen [2007]). However, to our knowledge, a fully developed semantic model of the CRM field is still missing. The main contribution of this paper is the proposal of a core reference ontology for the CRM domain, called O-CREAM-v2 (Ontology for Customer RElAtionship Management 2nd version), specifically targeted to small and medium sized enterprises. The main goal of O-CREAM-v2 is twofold: (a) providing a fully developed semantic model of the CRM field, aimed at fulfilling the needs mentioned above; (b) showing how a foundational ontology Borgo and Leitão [2004], providing an upper level model, enables the proper modeling of heterogeneous domains (such as the CRM one).

Our work started from an analysis of the CRM domain, aimed at individuating the main involed actors and eliciting the way in which they talk about CRM. This analysis, reported in Section 2.1, provided us with the basic requirements for building the CRM ontology, which are presented in Section 2.2. Section 3 discusses the reused upper level ontologies, while Section 4 provides a detailed presentation of O-CREAM-v2, which is a modified and extended version of O-CREAM (see Magro and Goy [2008a,b]). This section provides also a discussion of the most important differences between the first and the new version, mainly concerning the formalization of *relationships* and their *descriptions*. As mentioned above, many works can be found in the literature that are relevant to our work: in this section, along with the description of our proposal, the main related approaches are presented, and their role with respect to our work is explained. Section 5 concludes the paper by discussing possible exploitations of O-CREAM-v2.

2. Motivations, Domain Analysis and Requirements

2.1. Motivations and Domain Analysis

By an informal analysis of the adoption of CRM within Italian companies, it emerged that, if most of large companies have implemented some CRM strategies since years, many Small and Medium Sized Enterprises (SME) are still trying to orient themselves within this field. Italian SME, in particular, are traditionally very suspicious towards the adoption of new approaches and technologies, and tend to prefer going on using their traditional work methods (like hand-written notes, faxes, paper archives, and so on); meanwhile, in order to improve their competitiveness, they are forced to overcome their own resistance and take into consideration the adoption of new approaches to customer management, as well as ICT-based solutions that could support their business processes.

Given this scenario, we decided to focus our analysis on Italian SME, and we identified two possible roles a SME can play in relation with CRM:

- small ICT companies (i.e., software houses) can offer software solutions for CRM explicitely conceived for SME;
- SME can look for CRM software solutions, in order to improve their technological integration and business automation.

As already mentioned, we believe that a shared semantic model of the CRM field could be exploited as a common vocabulary, shared between ICT companies offering software solutions for CRM and SME looking for ICT support to their business activities. Moreover, it could be useful:

- as a basis for information integration and communications between business departments, cooperating towards CRM goals;
- as a support to the rationalization of a huge amount of knowledge that needs to be structured, elaborated and analyzed in order to be useful (e.g., for decision making);
- as a common reference meaning in all the computer mediated communications and computer supported activities concerning CRM.

Having in particular the mediation scenario in mind, the most important knowledge source for the elicitation of the requirements for such a shared semantic model of CRM is the way the involved actors talk about CRM within their business. Thus, we analyzed two main types of information sources:

- Documents (e.g., brochures and white papers), produced by ICT companies, describing their CRM tools.
- Interviews with
  - salesmen from ICT companies, aimed at eliciting the way they describe software solutions supporting CRM for SME;
  - managers of SME, in order to understand which concepts and terms they use to talk (and think) about their activities related to customer management.

Documents by ICT companies and interviews with their salesmen represent the way in which software solutions for CRM are described. Interviews with SME managers about CRM-related activities represent the way in which SME CRM activities and involved elements are described, as well as the way in which SME needs and requirements about technological support for CRM are expressed.

Since our goal was to build an ontology representing a common set of concepts and relations, in order to enable the mentioned actors to "talk" about CRM referring to shared meanings, we exploited both sources to elicit requirements for the CRM ontology.

Documents and interviews have been manually analyzed by individuating text fragments referring to CRM concepts. In particular, we grouped all together linguistic expressions used in documents/interviews and referring to the same concepts and relations. For instance, we found a set of expressions ("catalogazione documenti", "classificazione documenti", "archiviazione documenti", "archiviazione documen-

tale", "archivio documentazione", "archivio documenti"[1]) referring to the concept of *document classification* (see Axiom (A110) characterizing *DocumentClassification*). This analysis, applied to all documents and interviews, provided us with the following outcomes:

- A set of concepts and relations involved in the description of CRM activities in SME, i.e. concepts and relations that have to be characterized in a semantic model of the CRM domain.
- A list of natural language terms and expressions that emerged as important when talking about CRM; each term/expression is related to a semantic representation based on the ontology. These terms/expressions represent a bridge between the ontology and the user, by providing a natural language access to the semantic representation.

These outcomes have been checked, in order to find lacks or inaccuracies, by referring to the literature about CRM (e.g. Dychi [2001]; Freeland [2005]; Devalle [2005]), and by having them verified by a CRM expert. The former outcome provided us with the "content" requirements for the CRM ontology, that are summarized in the following. The latter outcome has been exploited to build a user interface enabling the users to interact with the ontology. Some preliminary results in this direction can be found in Goy and Magro [2011].

2.2. Requirements

We think that two kinds of requirements should be taken into account when building a domain ontology: First of all, there are *content requirements* deriving from the domain analysis; such requirements provide the knowledge engineer with the concepts and relations that the ontology should model. Moreover, there are *general requirements*, concerning general top-level concepts and their integration with domain specific concepts.

From the domain analysis, we elicited *content requirements*, stating which concepts and relations have to be characterized in a CRM ontology. Such concepts and relations can be grouped as follows:

1. Business activities (e.g., sales, offers, communications with customers). The analized documents refer to two basic kinds of business activities:
   - Activities somehow involving a business relationship. In turn, a business relationship is always a relation between the company and a stakeholder; for instance: sales, deliveries, payments, communications.
   - "Internal" activities, like filing documents, document (report, invoice, etc.) generation, updating (inserting, deleting) information in archives, and so on.
2. Relationships which the company is involved in (e.g., the relationship established between a vendor and a buyer after a sale). Relationships are those permanent links that are established usually as a result of a business activity.
3. The business knowledge of a company. Interesting examples are: reports about sales; calendars containing information about appointments; offer lists; archives and records about customers, sales, orders, and so on.
4. Software applications supporting business activities, with their properties (e.g., licences, functionality, technical requirements, delivery model).

From literature analysis about ontology modeling we elicited the following *general requirement*: a CRM ontology, suitable for supporting interoperability and communications both within or cross companies needs to be grounded on a foundational ontology Borgo and Leitão [2004], providing a rich characterization of the basic concepts and relations on which specific CRM concepts and relations can be formally defined.

The reuse of existing ontological models, and especially upper level ontologies, is of major importance in semantic knowledge representation, mainly because:

[1] Documents cataloguing, documents classification, documents filing, documentation filing, documentation file, documents file.

– Linking a domain ontology to an existing well-grounded upper level model guarantees a higher degree of accuracy in the definition of basic concepts.
– Making reference to the same upper level ontology enhances the interoperability between different domain ontologies.
– In domains characterized by a high heterogeneity of concepts, upper level ontologies provide a coherent framework supporting the management of such heterogeneity. In particular, the CRM domain is characterized by concepts and relations of very different nature, such as activities, information entities, software objects, and so on; linking the CRM domain ontology to a well-founded upper level ontology (see Section 3) helped us to cope with such heterogeneity, as we will show in details in Section 4.

3. Modeling Basis: Reused Ontologies

O-CREAM-v2 has been developed within the framework provided by the well-known foundational ontology *DOLCE* (Descriptive Ontology for Linguistic and Cognitive Engineering) Masolo *et al.* [2003]; Borgo and Masolo [2009], and three other ontologies Gangemi and Mika [2003]; Gangemi *et al.* [2005] that extend DOLCE, namely: the ontology of *Description and Situations (DnS)*, the *Ontology of Information Objects (OIO)*, and (to a lesser degree) the *Ontology of Plans (OoP)*.

In this section, we briefly sketch those features of DOLCE, DnS, OIO and OoP that are necessary for understanding O-CREAM-v2.

Other ontologies have influenced the development of O-CREAM-v2, which are not part of O-CREAM-v2 formalization. We do not provide here any survey of these latter ontologies, but we will mention them whenever it is needed to point out their similarities and differences with O-CREAM-v2.

In the following, the acronyms $DOLCE$, DnS, OIO and OoP are used to prefix the names of the concepts and relations of the respective ontologies, while the names of the elements of O-CREAM-v2 are written without prefix.

3.1. DOLCE

DOLCE Masolo *et al.* [2003]; Borgo and Masolo [2009] is a foundational ontology of particulars. $DOLCE : Particular$ denotes the root of the DOLCE taxomomy of categories, i.e. the class of all the particulars that DOLCE accounts for.

DOLCE distinguishes four basic categories of particulars, namely: *objects* ($DOLCE : Object$), *events* ($DOLCE : Event$), *qualities* ($DOLCE : Quality$) and *abstracts* ($DOLCE : Abstract$).[2]

Objects are those "particulars that are wholly present [...] at any time they are present" Masolo *et al.* [2003] (such as cars, human persons, sheets of paper, amounts of iron, legal persons, organizations, laws, data, etc.). Events are those entities that "happen in time" (such as activities, processes, etc.). Each object *participates in* at least one event (e.g., a human person participates in its life; when she sings, a singer participates to the singing activity, etc.); conversely, each event has at least one object that participates in it: $DOLCE : participant(x, y, t)$ specifies this immediate relation between objects and events, stating that *during time t, the object x participates in the event y*.

Among objects, it is worth mentioning the *physical objects* ($DOLCE : PhysicalObject$), such as cars, human persons, sheets of paper, amounts of iron, etc. and the *non-physical objects* ($DOLCE : NonPhysicalObject$), such as legal persons, organizations, laws, data, etc. *Physical objects* ($DOLCE : PhysicalObject$, such as cars, human persons, sheets of paper, etc.) and *amount of matter* ($DOLCE :$

[2] In Masolo *et al.* [2003] *objects* are called "endurants" and *events* are called "perdurants". Here we prefer the more business-friendly terms "objects" and "events", used in Borgo and Masolo [2009]. As a consequence, we also use the terms "physical/non-physical objects". Furthermore, the formalization in Borgo and Masolo [2009] is slightly different from that in Masolo *et al.* [2003]. We here mainly refer to the original DOLCE version described in Masolo *et al.* [2003], except for some minor aspects that are explicitly pointed out in this section.

$AmountOfMatter$, such as amounts of iron, etc.) are two special kinds of physical objects. The former "are objects with unity [, while the latter] are objects with no unity" Masolo *et al.* [2003]. The physical objects to which we ascribe some kind of agentivity are called *agentive physical objects* ($DOLCE : AgentivePhysicalObject$, such as human persons, etc.), while the others are called *non-agentive physical objects* ($DOLCE : NonAgentivePhysicalObject$, such as cars, sheets of paper, etc.). As a furhter category we have the one of *Non-physical (unitary) objects* ($DOLCE : NonPhysicalObject$). An important sub-category of non-physical objects is that of *social objects* ($DOLCE : SocialObject$), which somehow depend on a community of agents and among which DOLCE distinguishes between *agentive social objects* ($DOLCE : AgentiveSocialObject$) and *non-agentive social objects* ($DOLCE : NonAgentiveSocialObject$), according to whether or not we ascribe to them some kind of agentivity. Examples of the former are legal persons, organizations, etc., while examples of the latter are laws, data, etc.

Qualities are the "basic entities that we can perceive or measure" Masolo *et al.* [2003] (such as, colors, weights, etc.). In DOLCE, qualities are particulars and each individual quality *inheres in* exactly one particular, which is its own bearer (e.g., the *weight of John* is an individual quality uniquely pertaining to John and it is different from, say, *the weight of Mary*). $DOLCE : hasQuality(x, y)$ specifies that the particular x has the individual quality y (or, stated in other words, that the quality y inheres in the particular x). Physical objects can have only *physical qualities* ($DOLCE : PhysicalQuality$), while physical qualities can inhere in both physical objects and physical qualities themselves. Non-physical objects can have only *abstract qualities* ($DOLCE : AbstractQuality$), while abstract qualities can inhere in both non-physical objects and abstract qualities themselves. Events can have only *temporal qualities* ($DOLCE : TemporalQuality$), while temporal qualities can inhere in both events and temporal qualities themselves.

Each quality takes value in a $DOLCE : Region$. In Masolo *et al.* [2003], regions are special kinds of abstracts, which are particulars outside time and space. Differently, in Borgo and Masolo [2009], regions exist in time (i.e., they "are created, adopted, abandoned, etc" Borgo and Masolo [2009]): in the present article we take the latter perspective. There are three basic kinds of regions, namely the *physical regions* ($DOLCE : PhysicalRegion$), the *abstract regions* ($DOLCE : AbstractRegion$) and the *temporal regions* ($DOLCE : TemporalRegion$). Among the temporal regions, it is worth mentioning the *time intervals* ($DOLCE : TimeInterval$). Physical, abstract and temporal qualities take values in physical, abstract and temporal regions, respectively. As regards physical and abstract qualities, the predicate $DOLCE : qLocation(x, y, t)$, states that *during time t, the quality x is located in the region y.* A detailed discussion about the different approaches to modeling properties and the options offered by DOLCE (among which, the one based on individual qualities) can be found in Borgo and Masolo [2009].

Objects, events, qualities and, under the perspective taken in Borgo and Masolo [2009] and in the present paper, also regions are particulars that live in time: the predicate $DOLCE : presentAt(x, t)$ specifies that *during time t, the particular x exists.*

The parthood between objects is expressed by the predicates $DOLCE : part(x, y, t)$, which states that *y is part of x during t* and $DOLCE : properPart(x, y, t)$, which states that *during t, y is part of x, but x is not part of y* (i.e., that *during t, y is proper part of x*). Similarly, the parthood between events (and also between abstracts) is expressed by the predicates $DOLCE : part(x, y)$ and $DOLCE : properPart(x, y)$ (in the two former cases, the parthood relationship is not temporary). A particular x without proper parts is called an atom: $DOLCE : Atom(x)$.

In this article, we will exploit another basic DOLCE relation between particulars, namely the *specific constant dependence*: "A particular x is *specifically constantly dependent* on another particular y iff, at any time t, x can't be present at t unless y is also present at t" Masolo *et al.* [2003]. The predicate $DOLCE : specificallyConstantlyDependsOn(x, y)$ expresses this relation.

DOLCE reference formal characterization is specified in modal logic S5 plus the Barcan Formula, but lighter versions of DOLCE (also expressed in OWL W3C [2011]) exist as well and can be downloaded from the Web, e.g. from the Laboratory of Applied Ontology Web site (http://www.loa-cnr.it/) or the Ontology Design Patterns Web site (http://ontologydesignpatterns.org/wiki/Main_Page) .

3.2. Description and Situations (DnS)

The ontology of Description and Situations (DnS) Gangemi and Mika [2003]; Gangemi *et al.* [2005] provides a pattern for extending other ontologies with a set of reified concepts[3] and relations, which can represent tuples and the intensions or the extensions of classes and relations. The two core concepts are the notion of *description* ($DnS : Description$) and that of *situation* ($DnS : Situation$). Each instance of the latter represents "a state of affairs or relationship, a tuple, or fact" Gangemi *et al.* [2005]. Descriptions provide conceptualizations for (or give a structure to) situations and the basic relation between situations and descriptions is the satisfaction relation ($DnS : satisfies$) expressing the fact that a given situation satisfies a given description.

O-CREAM-v2 exploits, in particular, a DnS-based extension of DOLCE. However, in this last version of our ontology, we do not use the notion of $DnS : Situation$ (see Section 4.1.1 for a brief discussion about this issue) that we had used in a previous version Magro and Goy [2008b,a].

In such a DnS extension of DOLCE, *reified concepts*, i.e. the descriptive counterparts of particulars (e.g., the reification of the notions of examinee, taking an exam, exam duration, etc.), are special kinds of $DOLCE : NonAgentiveSocialObject$s (thus they are DOLCE particulars in their turn), characterized by the class $DnS : Concept$. The immediate relation between the reified concepts and the particulars that they make reference to is expressed by the predicate $DnS : classifies(x, y, t)$. DnS provides three special kinds of reified concepts: the *roles* ($DnS : Role$, such as the role of examinee), the *courses* ($DnS : Course$, such as the notion of taking an exam) and the *parameters* ($DnS : parameter$, such as the parameter specifying the exam duration) that represent the descriptive counterparts of $DOLCE : Object$s, $DOLCE : Event$s and $DOLCE : Region$s, respectively. Moreover, DnS provides also three relations that specialize the predicate $DnS : classifies(x, y, t)$, namely: $DnS : playedBy(x, y, t)$, holding between roles and the objects that they classify (e.g., the role of examinee, during a certain time, may be played by Mr. John Freeman); $DnS : sequences(x, y, t)$, holding between courses and the events that they classify (e.g., the course of taking an exam, during a certain time, may sequence the John Freeman's activity of taking that particular exam); $DnS : valuedBy(x, y, t)$, holding between parameters and the regions that they classify (e.g., the parameter specifying the exam duration can be valued by a region corresponding to 2 hours). At any time, a reified concept may classify one or more particulars or it may classify no particular at all. Conversely, at any time, a particular can be classified by one or more reified concepts or by no reified concept at all.

Each $DnS : Concept$ (and, thus, in particular, each $DnS : Role$, $DnS : Course$ and $DnS : parameter$) is always defined by a $DnS : Description$ ($DnS : defines(x, y)$ specifies that the description x defines the reified concept y) and, once defined, it can be used by any description ($DnS : uses(x, y)$ specifies that the description x uses the reified concept y). Defining is a special way of using, i.e.: $DnS : defines(x, y) \rightarrow DnS : uses(x, y)$.

OWL versions of DnS extensions of DOLCE can be found at URIs
http://www.loa-cnr.it/ontologies/ExtendedDnS.owl and http://ontologydesignpatterns.org/ont/dul/DUL.owl.

3.3. Ontology of Information Objects (OIO)

OIO Gangemi *et al.* [2005] further extends the DnS extension of DOLCE with a set of concepts, relations, definitions and axioms that account for information objects. In OIO, an information object ($OIO : InformationObject$) is a $DOLCE : NonAgentiveSocialObject$ and it represents an information content. OIO distinguishes both between an information and its physical realization ($OIO : InformationRealization$), and between an information and its encoding system ($OIO : InformationEncodingSystem$). A physical realization is any entity that acts as a physical support for an information (e.g. a paper sheet, a sound, a content of a sequence of cells in the main memory

[3] In the following, we use the words *concept*, *class* and *category* in an interchangeable way to refer to terminological elements of the ontology, while we use the expression "reified concept" to refer to those instances, belonging to the domain of discourse of the ontology, which represent concepts.

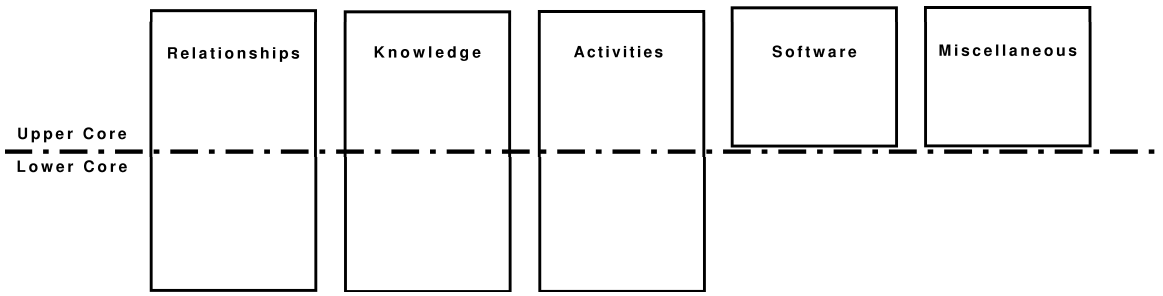

Fig. 1. O-CREAM-v2 modules and levels

of a computer, etc). The predicate $OIO : realizedBy(x, y, t)$ specifies that, *during time t, the information object x is realized by the information realization y*. Moreover, information contents are ordered according to some encoding systems, that is according to a set of rules (i.e. a language, a code, etc.), represented in the ontology by $OIO : InformationEncodingSystem$s, which are special kinds of $DnS : Description$s. The predicate $OIO : orderedBy(x, y)$ specifies that *the information object x is ordered by the information encoding system y*. Finally, during any time t, an information content x can express a meaning y, formalized as a $DnS : Description$ ($OIO : expresses(x, y, t)$), and can be about any particular, i.e. any element in the domain of discourse of the ontology, z ($oio : about(x, z, t)$). OWL versions of OIO are available at URIs http://www.loa-cnr.it/ontologies/InformationObjects.owl and http://ontologydesignpatterns.org/ont/dul/IOLite.owl.

3.4. Ontology of Plans (OoP)

The Ontology of Plans (OoP) Gangemi *et al.* [2005] accounts for several notions related to plans and their executions. We here will use only the general notion of *plan* ($OoP : Plan$), which is a special kind of description that "is conceived by a cognitive agent, defines or uses at least one task [...] and one role (played by agents), and has at least one goal as a proper part" Gangemi *et al.* [2005]. OWL versions of OoPcan be found at URIs http://www.loa-cnr.it/ontologies/Plans.owl and and http://ontologydesignpatterns.org/ont/dul/PlansLite.owl.

4. O-CREAM-v2: a core Ontology for Customer RElAtionship Management

An ontology for the CRM domain has to account for both *ground* particulars (such as activities, offers, sales, and other relationships, etc.) that we encounter in the domain and the information items about them. This is why several elements in O-CREAM-v2 representing ground particulars have also an informational counterpart in the ontology. Furthermore, since CRM is typically supported by ICT-based tools, a part of O-CREAM-v2 is devoted to the formal characterization of software applications.

O-CREAM-v2 is currently structured into five modules: Relationships, Knowledge, Activities, Software and Miscellaneous modules (Figure 1).

The first module provides a characterization for a basic notion of relationship and for the main kinds of business relationships in the CRM domain. The Knowledge module introduces the concept of $InformationElement$ as a refinement of $OIO : InformationObject$, as well as several kinds of information elements that play a central role in the considered domain. The third module characterizes a basic notion of activity and several more specific activity types. A notion of software and a basic framework for talking about several aspects of software applications are provided in the Software module. Finally, the Miscellaneous module accounts for a set of ontological items that support the formal apparatus of O-CREAM-v2, but are not central to the modeled domain: For these items, only a light characterization is provided.

Moreover, we distinguish two levels in O-CREAM-v2: an upper core and a lower core. The latter is more specific to the considered CRM domain, while the former is more general and can be useful also in

business domains other than CRM (Figure 1): Relationships, Knowledge and Activities modules span the two levels, while Software and Miscellaneous modules lie entirely within the upper core.

O-CREAM-v2 currently contains hundreds of classes, properties and axioms. Therefore, due to the space contraints we cannot provide here its complete formalization. Instead, in the present section we present its formal kernel, which allows one to understand the main aspects of the whole ontology. Furthermore, besides those notions for which a formal characterization is provided, we mention and informally describe also some other classes and properties, when we consider them useful to give a rather precise idea of O-CREAM-v2 coverage.

4.1. (Business) Relationships

Despite their very different nature, many states of affairs in the considered domain can be conveniently described as aggregations of interrelated particulars, in which each particular plays (one or more) specific roles. We call such aggregations *relationships*.

Throughout the article, we will use the term "relationship" both with a formal meaning (i.e., to refer to entities representing aggregations of particulars) and with the usual intended meaning in the CRM domain (e.g., in expressions as "business relationship", "sale relationship", etc.). Indeed, all of the different relationships in the domain share a common structure, which is effectively represented by an abstract formal notion of relationship.

For instance, when an enterprise sells a person some goods, a particular business relationship is established between the enterprise that sells and the person that buys. Moreover, such a relationship involves also other particulars, such as the sold goods and, possibly, other elements such as the specification of payment and delivery methods, etc. This "relationship", as intended in the CRM domain, can be effectively modeled as a "relationship" in the formal sense.

In Section 4.1.1, we introduce an abstract general formal notion of relationship. This formalization provides the basic machinery for "reifying" the relationships, i.e. to represent them as particulars and to place them in the domain of discourse. Such general notion of relationship encompasses not only the business relationships, but also all of those aggregation of interrelated entities that have to be predicated on within (and by means of) the ontology. For example, the association between a good and its asked price, which is not considered a business relationship in the CRM domain, it is formally modeled as a relationship.

In Section 4.1.2 we introduce some specializations of the general abstract notion of relationship and we there provide a characterization for the business relationships, as well as a description of some of the most relevant types of business relationships in the considered domain. Figure 2 reports a fragment of the O-CREAM-v2 Relationship Taxonomy (the names of classes of the relationships belonging to the upper core are written in bold), while Figure 3 lists some properties used to characterize the relationships (the names of those in the upper core are in bold) and depicts the hierarchy between properties (e.g., $hasSupplier$ is a subproperty of $hasRelationshipElement$).

4.1.1. (Business) Relationships: Upper Core

O-CREAM-v2 puts at disposal the basic general notion of $Relationship$, for representing the business relationships and, in general, all those relationships that need to be predicated on within the ontology.

$Relationship$ is the generic class of all the reified relation instances involving at least two elements (here called *relationships*); therefore, each subclass of $Relationship$ represents a reified n-ary relation (with $n \geq 2$). By means of their reification, the relationships are placed into the domain of discourse and thus they can be predicated on in a natural way within a first-order theory. Moreover, the reification of the relationships simplifies the translation of O-CREAM-v2 into those ontology languages that support only unary and binary relations, such as OWL W3C [2011].

The following six axioms provide the basic minimal formal machinery for representing relationships.

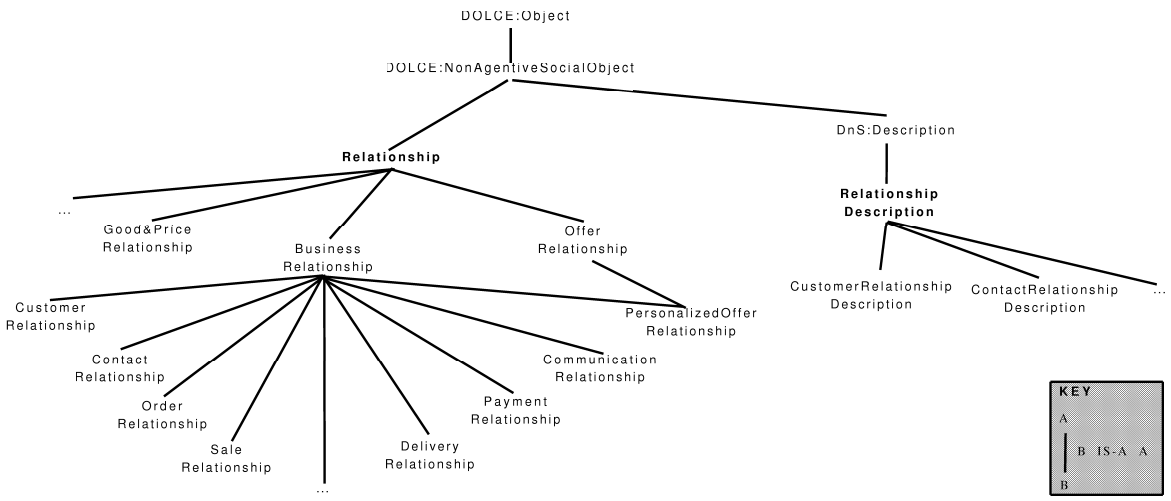

Fig. 2. A fragment of O-CREAM-v2 Relationship Taxonomy

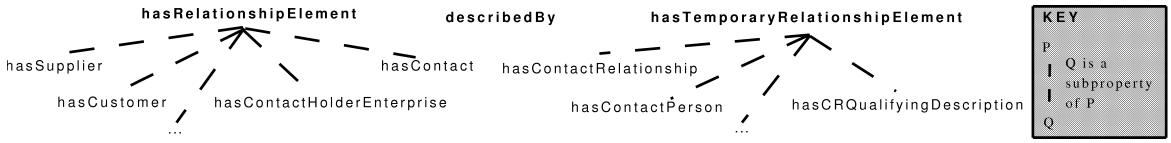

Fig. 3. Most relevant properties characterizing O-CREAM-v2 relationships

Each relationship involves at least two individuals (elements of the relationship) and it is described by a $RelationshipDescription$, which is a particular type of $DnS : Description$ that specifies the classifying reified concepts for the individuals in the relationship:[4]

(A1) $Relationship(x) \rightarrow DOLCE : NonAgentiveSocialObject(x) \wedge$
$(DOLCE : presentAt(x,t) \rightarrow (\exists y1, y2, y3)(y1 \neq y2 \wedge$
$(hasRelationshipElement(x, y1) \vee hasTemporaryRelationshipElement(x, y1, t)) \wedge$
$(hasRelationshipElement(x, y2) \vee hasTemporaryRelationshipElement(x, y2, t)) \wedge$
$RelationshipDescription(y3) \wedge describedBy(x, y3)))$

The association between a relationship and an element of the relationship can be either temporary or not. In some cases, a relationship can acquire and lose some elements during its life span, still maintaining its identity. This is true, in particular, for many business relationships in the considered domain, which may change in time. For instance, in the same contact relationship between an enterprise e (the "contact holder") and another enterprise or person c (the "contact"), the (optional) person that plays the role of contact point for the communications within the relationship may change during the life span of the relationship, without affecting the identity of the relationship, which thus remains (i.e., it continues to be regarded as) the same contact relationship between e and c (see Section 4.1.2). In order to express this kind of association between a relationship and an element, we introduce the following temporary property:

(A2) $hasTemporaryRelationshipElement(x, y, t) \rightarrow$
$Relationship(x) \wedge DOLCE : Particular(y) \wedge DOLCE : TimeInterval(t)$

In other cases, the association between a relationship and an element is constant, i.e. it holds for the whole life span of the relationship. For instance, in a contact relationship, the contact holder enterprise e and the contact c can not change during the life span of the relationship, since the identity of the relation-

[4]Every formula is assumed to be universally quantified over all its free variables. We specify axioms, definitions and theorems in a first-order language with equality, thus the equality symbol "=" is a primitive logical symbol; "$x \neq y$" is a shortcut for "$\neg(x = y)$".

ship depends on the involvement of both e and c in the relationships (i.e., a relationship involving another contact holder in place of e or another contact in place of c would be another relationship). The following non temporary property expresses such kind of association between a relationship ad an element:

(A3) $hasRelationshipElement(x, y) \rightarrow Relationship(x) \wedge DOLCE : Particular(y)$

A relationship is always described by a description that specifies the roles that the elements play within the relationship. We firstly introduce the general predicate $describedBy(x, y)$, whose intuitive meaning is that *the particular x is described by the description y*:

(A4) $describedBy(x, y) \rightarrow DOLCE : Particular(x) \wedge DnS : Description(y)$

We do not provide a formalization for the $describedBy$ property in the general case, but we restrict it in the case of the relationship descriptions by stating that any relationship description, for each element of the relationship that it describes, must at least *use* ($DnS : uses$) one reified concept that classifies that element:

(A5) $RelationshipDescription(x) \rightarrow$
$DnS : Description(x) \wedge (\forall y)(describedBy(y, x) \rightarrow Relationship(y))$

(A6) $Relationship(x) \wedge RelationshipDescription(y) \wedge describedBy(x, y) \wedge$
$(hasRelationshipElement(x, z) \vee hasTemporaryRelationshipElement(x, z, t)) \wedge$
$DOLCE : presentAt(x, t) \wedge DOLCE : presentAt(y, t) \rightarrow$
$(\exists c)(DnS : Concept(c) \wedge DnS : uses(y, c) \wedge DnS : classifies(c, z, t))$

The notion of $DnS : Situation$ and, in general, the situation-description framework of DnS Gangemi *et al.* [2005] has influenced our formal characterization of the concept of $Relationship$. Indeed, a first version of the ontology Magro and Goy [2008b,a] specified the relationships as particular $DnS : Situations$ and the relation between a relationship and its description was modeled by the $DnS : satisfies$ property, holding between situations and descriptions.

Even though several good reasons had supported such a choice, in this new version of the ontology we revised it, since the notion of $DnS : Situation$ revealed itself not very suited to our representation needs (as a consequence, the primitive property $describedBy$ has been introduced and used to link the relationships to their descriptions, in place of $DnS : satisfies$). For instance, the need to distinguish between temporary and non temporary elements in a relationship can not be easily satisfied by that framework, where only the non temporary property $DnS : settingFor$ is available to represent the association between a $DnS : Situation$ and the individuals that it involves. Furthermore, no $DnS : Situation$ can be a setting for another $DnS : Situation$ and each $DnS : Situation$ is always a setting for at least one $DOLCE : Event$ (see definition $D5$ and axioms $A10$ and $A11$ in Gangemi *et al.* [2005]): Such a characterization seems too restrictive in the considered CRM domain.

4.1.2. (Business) Relationships: Lower Core

The enterprises are daily involved in many *business relationships* and those that want to implement some kind of CRM are primarily interested in *managing* these relationships. This means that they need to keep track of them in their information systems, to maintain data and information about them, to derive new knowledge from those data and information, to (reactively or proactively) establish new business relationships, etc. In this domain, a business relationship is any relationship that an enterprise has with a stakeholder (i.e., a customer, a contact, a buyer, etc.). In this context, several different kinds of state of affairs are regarded as business relationships: besides the customer relationships, which are complex relationships involving other relationships (see the end of this section), also personalized offers, orders, sales, communications, as well as the simple fact that an enterprise is in contact with some other enterprise or human person, are all examples of business relationships.

Formally, a business relationship is defined as a relationship constantly (i.e., by means of the non temporary $hasRelationshipElement$ property) involving at least an enterprise e and another individual (organization or human person) sh,[5] which plays some stakeholder role within the relationship:

(D1) $BusinessRelationship(x) \equiv$
$Relationship(x) \wedge (\forall t)(DOLCE : presentAt(x,t) \rightarrow$
$(\exists e, sh, shr, d)(Enterprise(e) \wedge (Organization(sh) \vee HumanPerson(sh)) \wedge$
$StakeHolderRole(shr) \wedge RelationshipDescription(d) \wedge e \neq sh \wedge$
$hasRelationshipElement(x,e) \wedge hasRelationshipElement(x,sh) \wedge$
$DnS : playedBy(shr, sh, t) \wedge DnS : uses(d, shr) \wedge describedBy(x,d)))$

A stakeholder role is a particular kind of $DnS : Role$ and it can be played only by organizations or human persons:

(A7) $StakeHolderRole(x) \rightarrow DnS : Role(x) \wedge$
$(\forall y, t)(DnS : playedBy(x, y, t) \rightarrow (Organization(y) \vee HumanPerson(y)))$

Some particular types of stakeholder roles have been specified in the ontology, such as the roles of contact, buyer, customer, etc.

O-CREAM-v2 accounts for several core business relationship types in the CRM domain. Here we describe only some of them in order to illustrate how they are characterized within the framework described so far. In particular, in the following, we will provide a formal specification only for the notion of *contact relationship*, with the purpose of illustrating the formal pattern that we have followed for characterizing relationships within O-CREAM-v2. Then, we will informally describe the core relationship types involved in the sales cycle (namely: the (possibly *personalized*) *offer*, *order*, *sale*, *delivery*, *payment* and *good-price association* relationhip types), as well as the notion of *customer relationship*.

When an enterprise is in touch with some stakeholder, a *contact relationship* holds between them. The stakeholder might have bought some products or services from the enterprise or it might buy something in the future. In any case, the supplier is interested in selling its product or services to the customer.

We characterize the contact relationship as a particular business relationship whose elements may play different roles, namely: the contact holder enterprise, the contact, the contact person, and the qualifying description roles. These roles are used in a *contact relationship description*, as stated by the following axiom:

(A8) $ContactRelationshipDescription(x) \rightarrow$
$RelationshipDescription(x) \wedge (\exists ch, cr, cpr, qdr)(ContactHolderEnterpriseR(ch) \wedge$
$ContactR(cr) \wedge ContactPersonR(cpr) \wedge CRQualifyingDescriptionR(qdr) \wedge$
$DnS : uses(x, sr) \wedge DnS : uses(x, cr) \wedge DnS : uses(x, cpr) \wedge DnS : uses(x, qdr))$

Since we aim at modeling the contact relationships between enterprises and their contacts, only an enterprise can be the contact holder:

(A9) $ContactHolderEnterpriseR(x) \rightarrow DnS : Role(x) \wedge$
$(\forall y, t)(DnS : playedBy(x, y, t) \rightarrow Enterprise(y))$

Differently, the contact role is a stakeholder role and thus it can be played both by organizations (thus, in particular, by enterprises) and human persons:

(A10) $ContactR(x) \rightarrow StakeHolderRole(x)$

In a contact relationship, there can be some persons that the contact holder enterprise actually refers to for communicating with its contact:

[5] The notions of *organization*, *enterprise* and *human person* are beyond the scope of our ontological investigation. The Miscellaneous module (Section 4.5) contains only the specification that $Organization$ is a $DOLCE : AgentiveSocialObject$, $Enterprise$ is a $Organization$ and $HumanPerson$ is a $DOLCE : AgentivePhysicalObject$.

(A11) $ContactPersonR(x) \rightarrow DnS : Role(x) \wedge$
$(\forall y,t)(DnS : playedBy(x,y,t) \rightarrow HumanPerson(y))$

Finally, a contact relationship may be further qualified by means of some descriptions that can specify, for instance, the preferred ways of contact, payment or delivery methods for the customer:

(A12) $CRQualifyingDescriptionR(x) \rightarrow DnS : Role(x) \wedge$
$(\forall y,t)(DnS : playedBy(x,y,t) \rightarrow DnS : Description(y))$

A *contact relationship* is a business relationship described by a contact relationship description:

(D2) $ContactRelationship(x) \equiv Relationship(x) \wedge$
$(\exists y)(ContactRelationshipDescription(y) \wedge describedBy(x,y))$

(A13) $ContactRelationship(x) \rightarrow BusinessRelationship(x)$

For each role type specified above, there is a corresponding property that links the relationship with the elements that play this kind of role within the relationship itself. Thus, a contact relationship is characterized by the following properties: $hasContactHolderEnterprise(x,y)$, $hasContact(x,y)$, $hasContactPerson(x,y,t)$, and $hasCRQualifyingDescription(x,y,t)$.
For the two former properties, an axiom like the following three holds:

(A14) $hasContactHolderEnterprise(x,y) \rightarrow$
$ContactRelationship(x) \wedge hasRelationshipElement(x,y)$

Each contact relationship involves one and only one particular contact holder and one and only one particular contact:

(A15) $ContactRelationship(x) \rightarrow (\exists y)(hasContactHolderEnterprise(x,y))$

(A16) $hasContactHolderEnterprise(x,y1) \wedge hasContactHolderEnterprise(x,y2) \rightarrow y1 = y2$

For the other two properties $hasContactPerson$, and $hasCRQualifyingDescription$, we do not specify any axiom similar to the two latter, since we admit both that in a contact relationship there can be no contact person (or qualifying descriptions) and that there can be one or more. Moreover, the set of contact persons (or of qualifying description) in a contact relationship may change during the life span of the relationship, therefore these are two temporary properties:

(A17) $hasContactPerson(x,y,t) \rightarrow$
$ContactRelationship(x) \wedge hasTemporaryRelationshipElement(x,y,t)$

(analogously for $hasCRQualifyingDescription(x,y,t)$).
Each contact relationship depends both on the contact holder and on the contact, therefore if any of them ceases to be present, the relationship ceases to be present too:

(A18) $ContactRelationship(x) \wedge hasContactHolderEnterprise(x,y) \rightarrow$
$DOLCE : specificallyConstantlyDependsOn(x,y)$

(analogously for $hasContact(x,y)$)
The following axiom specifies that a contact holder element in a contact relationship always plays a contact holder role, which is used by a corresponding contact relationship description (similar axioms hold for the other above listed three properties that characterize the contact relationships):

(A19) $ContactRelationship(x) \wedge ContactRelationshipDescription(y) \wedge describedBy(x,y) \wedge$
$hasContactHolderEnterprise(x,z) \wedge$
$DOLCE : presentAt(x,t) \wedge DOLCE : presentAt(y,t) \rightarrow$
$(\exists ch)(ContactHolderEnterpriseR(ch) \wedge DnS : uses(y,ch) \wedge DnS : playedBy(ch,z,t))$

In the CRM domain, the relations typically involved in the sales cycle are of paramount importance: We provide in the following an informal characterization for the principal ones.

The $\underline{OfferRelationship}$ is a first central relation in the sales cycle. Any offer relationship is a $Relationship$ amongst the $Enterprise$ that makes the offer, one or more offered goods (each with its asked price), one or more (optional) targets, a $DOLCE : TimeInterval$ specifying the validity time, an optional global asked price (i.e. the price asked for the set of all the offered goods), and one or more (optional) qualifying descriptions (that may specify, for instance, the possible payment or delivery methods). As far as the offered goods are concerned, each $OfferRelationship$ always has one or more $\underline{Good\&PriceRelationship}$ elements; each of them is a $Relationship$ representing the association between an offered good, its asked price and/or its asked unit price, and a set of descriptions qualifying that single offered good.

As argued in Hepp [2008], an offered good may be either an actual, concrete "identifiable object or action" (such as that particular house or that specific tutorial that prof. X will give at that time at a given place) or a characterization of a set of similar goods (e.g., by means of the specification of their common make and model, such as in an offer for *PCs Dell Latitude E6400*), which only specifies the type of the offered goods without explicitly referring to any specific "identifiable object or action". Therefore, an offered good may be either a specific $DOLCE : Particular$ (or a set of $DOLCE : Particulars$), which is the very offered element, or a $DnS : Description$ that characterizes the type of the offered goods (and, possibly, their quantity or quantity range). In this last case, the $Good\&PriceRelationship$ contains a $DnS : Description$ that characterizes the offered goods, but it makes no direct reference to the corresponding actual individual goods. However, since at least one individual corresponding to the specified type should exist, we require that the $DnS : Description$ of the offered goods always $DnS : uses$ a concept that $DnS : classifies$ at least one concrete individual (or set of individuals). For instance, in an offer for *Dell Latitude E6400* PCs, we require that at least one Dell Latitude E6400 concrete PC exists that corresponds to the description.

For the sake of generality, we admit that any kind of individual can be offered. However, typically an offered good is a $DOLCE : AmountOfMatter$, a $DOLCE : NonAgentivePhysicalObject$[6], a $Service$, an $InformationElement$, an $AmountOfMoney$ (in financial transactions), a set of elements of these types or a $DnS : Description$, which $DnS : uses$ a concept that $DnS : classifies$ some element of these types.[7]

As done with the type of the offered goods, we do not place any restriction on the nature of the prices. Usually, a price is a $DnS : Description$ representing the specification of an $AmountOfmoney$, but, in general, everything can be a price. Therefore, $OfferRelationship$ encompasses those situations (that should be included, although they are not very common) in which a product or a service is offered for a change with another product or service. Thus, it should be clear that we admit a sort of symmetry between goods and prices, which somehow holds also in other relationships defined in the following. Indeed, in principle, every individual that can be offered as a good can also be asked as a price.[8]

Finally, an offer relationship can specify one or more targets which the offer is addressed to. Any target in an $OfferRelationship$ can be a specific $Organization$ (e.g., the ACME, Inc.), a specific $HumanPerson$ (e.g., John Smith) or a $DnS : Description$ characterizing the actual targets.

In many cases, the CRM practices aim at personalizing the offers in order to meet as much as possible the needs of the individual customers: a $\underline{PersonalizedOfferRelationship}$ is an $OfferRelationship$ and a $BusinessRelationship$ (since it directly involves a stakeholder) that has one and only one specific target, which is either a specific $Organization$ or a specific $HumanPerson$.

[6] According to the notion of product specified in Borgo and Vieu [2006], a product always is an individual in $DOLCE : AmountOfMatter \cup DOLCE : NonAgentivePhysicalObject$.

[7] The notions of *service* and *amount of money* are beyond the scope of our ontological investigation. The Miscellaneous module (Section 4.5) contains only the specification that $Service$ and $AmountOfMoney$ are two kinds of $DOLCE : Particulars$.

[8] Such a symmetry is recognized also by other authors: For instance, in Uschold *et al.* [1998], the *product* (here called *good*) is defined as "the role of the good, service or quantity of money that is: offered for sale by a vendor or agreed to be exchanged by the vendor with the actual customer in a sale", while the *asking price* is defined as: "the role of the good, service or quantity of money being asked for by a vendor in exchange for a product that is for sale" (an analogous definition is provided in Uschold *et al.* [1998] also for the *sale price*).

Everything stated above about the goods and the prices in an offer relationship holds also for the goods and the prices in the order and sale relationships, described in the following.

Each time an enterprise receives an order, a particular $\underline{OrderRelationship}$ is established. Any order relationship is a $BusinessRelationship$ amongst the $Enterprise$ that receives the order, the $Organization$ or the $HumanPerson$ that places the order, one or more ordered goods (each with its price), a $DOLCE : TimeInterval$ representing the date of the order, an optional global order price (i.e., the price that has to be paid for the set of all the ordered goods) and one or more (optional) qualifying descriptions. Like the $OfferRelationship$s, any $OrderRelationship$ can specify several ordered goods, each one represented by a $Good\&PriceRelationship$.

Similarly, when something is sold by a vendor to a buyer, a $\underline{SaleRelationship}$ is created. Every individual sale relationship is a $BusinessRelationship$ amongst the vendor $Enterprise$, the buyer $Organization$ or $HumanPerson$, one or more sold goods (each with its price), a $DOLCE : TimeInterval$ representing the date of the sale, an optional global sale price (i.e., the price that has to be paid for the set of all the sold goods) and one or more (optional) qualifying descriptions. Any $SaleRelationship$ can specify several sold goods, each one represented by a $Good\&PriceRelationship$.

Finally, we briefly characterize the delivery and payment relationship types.

A $\underline{DeliveryRelationship}$ represents the delivery of some goods to a buyer. Its elements are the delivering $Enterprise$, the $Organization$ or the $HumanPerson$ that receives the goods, one or more delivered goods (or collections of goods), an optional delivery charge (i.e., the price that has to be paid by the receiver for the delivery), a delivery channel (e.g., the postal service, a courier service, the Internet, etc.) and a $DOLCE : TimeInterval$ representing the delivery date. It is worth pointing out that actual, concrete goods are delivered. Thus, differently from the offer, order and sale relationships, the delivery relationships always concern only actual, concrete goods and not their characterizations by means of $DnS : Description$s.

A $\underline{PaymentRelationship}$ represents the transfer of a paid price from a payer to a paid enterprise. Its elements are the $Enterprise$ that receives the payment, the $Organization$ or the $HumanPerson$ that pays the price, the paid price, an optional charge (i.e., the additional price that has to be paid by the payer for that specific payment method), an optional payment channel and a $DOLCE : TimeInterval$ representing the payment date. It is worth pointing out that a paid price always is an actual concrete element (or a collection or actual concrete elements), such as an $AmountOfMoney$. Thus, differently from the offer, order, sale and delivery relationships, the payment relationships always concern actual concrete elements as paid price and payment charges and not their characterizations by means of $DnS : Description$s.

Besides the relationships described above, O-CREAM-v2 also accouts for several other relationship types relevant in the CRM domain, among them: the *request for quotation*, *bidding*, *complaint*, *appointment*, *visit*, *communication* and *conversation* (with contacts) business relationship types.

The various business relationships between an enterprise and a stakeholder (personalized offers, orders, sales, payments, communications, etc.) characterize that complex relationhip holding between them and known as *customer relationship*. Therefore O-CREAM-v2 formalizes the notion of $\underline{CustomerRelationship}$ as a (complex) $BusinessRelationship$ amongst an $Enterprise$ (the *supplier*), an $Organization$ or a $HumanPerson$ (the *customer*, either potential or actual) and a set of other business relationships ($ContactRelationship$, $OfferRelationship$s, $SaleRelationship$s, $PaymentRelationship$s, $CommunicationRelationship$s, etc.), each one involving both the same enterprise (which plays the supplier role in the customer relationship) and the same organization or human person (which plays the customer role in the customer relationship).

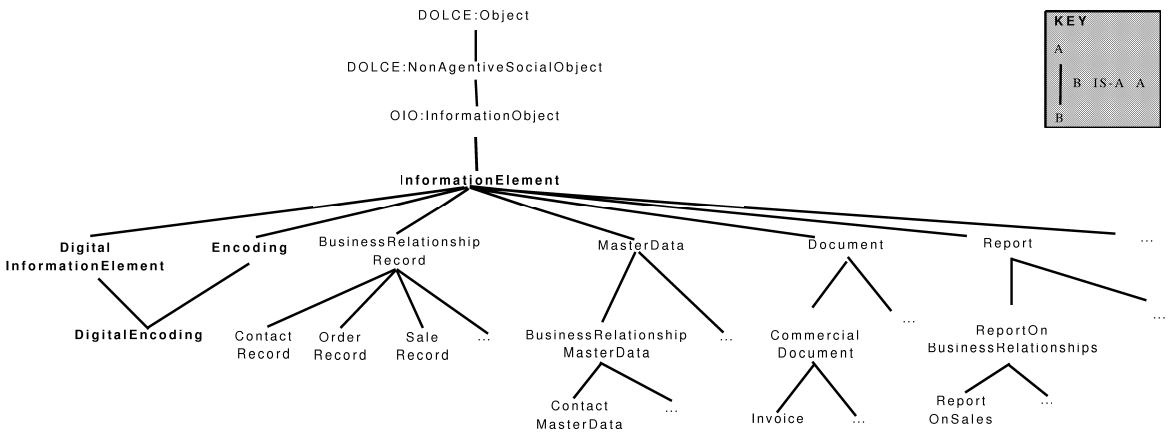

Fig. 4. A fragment of O-CREAM-v2 Information Element Taxonomy

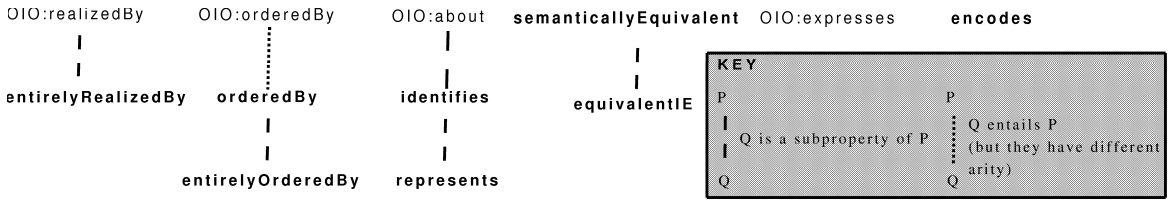

Fig. 5. Most relevant properties characterizing O-CREAM-v2 information elements

4.2. (Business) Knowledge

The CRM activities manage a huge amount of business knowledge[9]: They access the master data files, the transactional data and various information sources, they analyze data and information, they produce data, documents and reports, they register in databases new contacts, orders, sales, communications, etc.

The Ontology of the Information Objects Gangemi *et al.* [2005] provided us with the basis for formalizing the concepts and the relations relevant to the business knowledge. O-CREAM-v2 introduces the notion of $InformationElement$ as a sub-concept of $OIO : InformationObject$ and it strengthens for it several OIO axioms that characterize the latter, in order to better fit the considered domain. Moreover, a set of new properties (w.r.t. OIO) are introduced, formalized and related to those in OIO.

Figure 4 shows a fragment of the O-CREAM-v2 taxonomy of information elements, while Figure 5 reports the most relevant properties that characterize the information elements (as in the figures above, the names of the classes and properties belonging to the upper core are written in bold). In Figure 5, a dashed line expresses a sub-property/super-property relationship, while a dotted line represents an entailment relationship between two properties that formally is not a sub-property/super-property relationship. For instance, $entirelyOrderedBy$ is a sub-property of $orderedBy$, while the latter (which has arity 3) is not a sub-property of $OIO : orderedBy$ (which has arity 2); however, it holds that $(\forall x, y, t,)(orderedBy(x, y, t) \rightarrow OIO : orderedBy(x, y))$.

In Section 4.2.1, we provide a characterization for the general $InformationElement$ concept by refining and extending the OIO framework. Moreover, we also informally introduce the notion of *digital information element* and the $encodes$ property, which relates an encoding information element to its encoded information. All of them have been placed in the upper core of the ontology, given their possible utility also in several other domains different from CRM. In Section 4.2.2 we introduce and discuss some more specific kinds of information elements, with the main aim of illustrating how the basic characterization of information elements applies to them.

[9]We refer here to a broad notion of *business knowledge* encompassing all kinds of data, information and knowledge somehow involved in the CRM business processes.

4.2.1. (Business) Knowledge: Upper Core

The characterization of the data, information and knowledge elements involved in the (CRM) business processes requires at least the capability of talking about the meaning, the encoding languages and the physical realizations of such elements. As stated in Section 3.3, OIO formalizes the basic notion of information object ($OIO : InformationObject$) by considering all these features. In particular, OIO offers a framework in which the content of an information object is kept clearly separate both from its physical realization and from its encoding system and, furthermore, in which the relationships between these notions are analyzed and represented, as well. In O-CREAM-v2, the concepts and the relations related to the business knowledge are rooted in that framework. However, although the notion of $OIO : InformationObject$ is the foundation for our formalization, it is too underspecified to be directly used in the CRM domain. For this reason, we have introduced $InformationElement$ as a sub-concept of $OIO : InformationObject$ and we have refined and extended the characterization that the former inherits from the latter.

The main aims of this refinement and extension are:

1. Clarifying the relationships between an information element and its parts for what it concerns their physical realizations, their encoding systems, the particulars they are about, and the descriptions they express. For instance, if a part of a document is realized by means of a sequence of paper sheets, what can we say about the physical realization of the whole document? Conversely, if a document is physically realized by a sequence of paper sheets, what can we say about its proper parts? If all of the proper parts of a document are physically entirely realized by the same sequence of paper sheets, what can we say about the whole document? Conversely, if a document is entirely realized by a sequence of paper sheets, what can we say about its proper parts? Similar questions in relation to the encoding systems, the particulars that an information element is about and the descriptions that it expresses have to be answered too.
2. Clarifying the relationships between an information physical realization and its possible parts that are physical realizations in their turn, for what it concerns the information elements that they realize. For instance, if a sequence of ten paper sheets (partially/entirely) physically realizes a document, what can we say about a booklet that contains those ten paper sheets? Can we say that the booklet (partially/entirely) physically realizes the document too? If each proper part of a document is entirely physically realized by a sequence of pages of a book, can we say that the book, as a whole, entirely realizes the document?
3. Providing the means to clearly decouple the encoding systems of an information element from its physical realizations. This will allow, for instance, to unambiguosly specify that some business knowledge piece exists both in an English and in an Italian version and that the first one is on paper, while for the second one only a digital encoding exists.
4. Clarifying the relationships between a description and its parts for what it concerns the information elements that express them; i.e., does an information element that expresses a description also expresses the possible descriptive parts of the expressed description?
5. Accounting for the semantic equivalence between information elements and for a stricter equivalence notion encompassing, not only the equivalence between the meanings of two information elements, but also between the encodings that order them. This means, in particular, that we admit that two information elements can be equivalent (both semantically and, possibly, also with respect to their ordering encoding systems) still being two distinct individuals.
6. Accounting for those information elements that *identify* or *represent* particulars (different from themselves), within an enterprise information system.

In the following, we present and discuss our formal framework.

$InformationElement$ is the most general concept that represents the set of all the information elements in our domain:

(A20) $InformationElement(x) \rightarrow OIO : InformationObject(x)$

Every part of an information element is an information element:

(A21) $InformationElement(x) \wedge DOLCE : part(x, y, t) \rightarrow InformationElement(y)$

In general, an individual information element (e.g., the specification of the best practices for handling some specific kinds of cutomer's complaints) can be physically realized by different elements. We need to distinguish the situations in which each realizing element realizes the whole information (e.g., such best practices are wholly described in each booklet given to each employee working in the marketing unit) from those in which each realizing element only partially realizes the information (e.g., the best practices are partly decribed in a booklet and partly present only in the mind of the employee in charge at the marketing unit). Therefore, for each information element x, we interpret the predicate $OIO : realizedBy(x, y, t)$ as the assertion that, during t, x is physically realized by y, *either entirely or only partially*; furthermore, we introduce the predicate $entirelyRealizedBy(x, y, t)$ to express the fact that, during t, x is entirely physically realized by y (for instance, in the first above-mentioned situation, the best practices are $entirelyRealizedBy$ each booklet; in the second one we can only say that they are $OIO : realizedBy$ both the booklet and an employee's mental representation, but no realization entirely physically realizes the information element. However, we can specify that a proper part of the best practices is $entirelyRealizedBy$ a booklet and another one is $entirelyRealizedBy$ the employee's mental realization).

Formally, we first state that if a part of an information element is realized by any physical realization, then the whole information element is realized by that physical realization too:

(A22) $InformationElement(x) \wedge DOLCE : part(x, z, t) \wedge OIO : realizedBy(z, y, t) \rightarrow$
$OIO : realizedBy(x, y, t)$

Stating that an information element is realized by a physical realization means that either it is entirely realized by such a physical realization, or it has a proper part that is realized by it:

(A23) $InformationElement(x) \wedge OIO : realizedBy(x, y, t) \rightarrow$
$(entirelyRealizedBy(x, y, t) \vee$
$(\exists z)(DOLCE : properPart(x, z, t) \wedge entirelyRealizedBy(z, y, t)))$

The property $entirelyRealizedBy(x, y, t)$ is stronger than $OIO : realizedBy(x, y, t)$:

(A24) $entirelyRealizedBy(x, y, t) \rightarrow$
$InformationElement(x) \wedge OIO : realizedBy(x, y, t)$

If a physical realization entirely realizes an information element, then it entirely realizes also any proper part of that information element:

(A25) $entirelyRealizedBy(x, y, t) \rightarrow$
$(\forall z)(DOLCE : properPart(x, z, t) \rightarrow entirelyRealizedBy(z, y, t))$

Conversely, if a physical realization entirely realizes each proper part of a non-atomic information element, then it entirely realizes also the whole information element:

(A26) $InformationElement(x) \wedge DOLCE : properPart(x, z1, t) \wedge$
$(\forall z)(DOLCE : properPart(x, z, t) \rightarrow entirelyRealizedBy(z, y, t)) \rightarrow$
$entirelyRealizedBy(x, y, t)$

Up to here we have formalized the relationships between an information element and its parts for what it concerns the physical realizations. By taking the opposite perspective, we need to specify the relationships between an information physical realization and its parts for what it concerns the realized information elements. For instance, if a section of a booklet physically realizes an information element (e.g., the above-mentioned best practices), then also the booklet as a whole physically realizes that information element. In general, if an information element x is realized by an information realization y, then it is also realized by any information realization that has y as a part, as stated by the following axiom (in which both the

temporary and the non temporary partonomic relations occur, since an information realization can be either an object or an event):

(A27) $InformationElement(x) \wedge OIO : realizedBy(x, y, t) \wedge$
$OIO : InformationRealization(z) \wedge (DOLCE : part(z, y, t) \vee DOLCE : part(z, y)) \rightarrow$
$OIO : realizedBy(x, z, t)$

An analogous axiom holds also for the entire realization:

(A28) $InformationElement(x) \wedge entirelyRealizedBy(x, y, t) \wedge$
$OIO : InformationRealization(z) \wedge (DOLCE : part(z, y, t) \vee DOLCE : part(z, y)) \rightarrow$
$entirelyRealizedBy(x, z, t)$

Given that a proper part of an object is also a part of that object, from Axioms $A21$, $A26$ and $A28$, it follows that if each proper part of a non atomic information element x is entirely realized by any information physical realization, which is part of a given information physical realization w, then w entirely realizes the whole information element x:

(T1) $InformationElement(x) \wedge OIO : InformationRealization(w) \wedge$
$DOLCE : properPart(x, z1, t) \wedge (\forall z)(DOLCE : properPart(x, z, t) \rightarrow$
$(\exists y)(entirelyRealizedBy(z, y, t) \wedge DOLCE : part(w, y, t)) \rightarrow$
$entirelyRealizedBy(x, w, t)$

Similarly to what happens with the physical realizations, the same individual information element (e.g., a sales report) can be ordered by different encoding systems. We need to distinguish the situations in which each encoding system orders the whole information (e.g., the report is entirely written both in Italian and in English) from those in which each encoding system only partially orders the information (e.g., the report is partly written in English and partly specified in a diagrammatic language). Moreover, the set of encoding systems that order the same information element may change during the life span of the information element (e.g., the report is originally produced in Italian and, afterwards, it is translated in English). This last dynamic behaviour of the information elements w.r.t. their encoding systems can not be expressed by the non temporary $OIO : orderedBy(x, y)$ relation specified in Gangemi *et al.* [2005], however, we easily overcome this obstacle (while still committing to OIO) by introducing the temporary $orderedBy(x, y, t)$, as follows:

(A29) $orderedBy(x, y, t) \rightarrow InformationElement(x) \wedge$
$OIO : InformationEncodingSystem(y) \wedge DOLCE : TimeInterval(t)$

(A30) $orderedBy(x, y, t) \rightarrow OIO : orderedBy(x, y)$

(A31) $OIO : orderedBy(x, y) \rightarrow (\exists t)(orderedBy(x, y, t))$

We interpret the predicate $orderedBy(x, y, t)$ as the assertion that, during time t, x is ordered by y, *either entirely or only partially*; furthermore, we introduce the predicate $entirelyOrderedBy(x, y, t)$ to express the fact that, during t, x is entirely ordered by y (for instance, in the first above-mentioned situation, the report is $entirelyOrderedBy$ the Italian language and $entirelyOrderedBy$ the English language, as well; in the second one we can only state that it is $OrderedBy$ both the English language and a diagrammatic language, but none of these languages entirely orders the report. However, we can specify that a proper part of the report is $entirelyOrderedBy$ the English language and another one is $entirelyOrderedBy$ a diagrammatic language).

Formally, we first state that if a part of an information element is ordered by any information encoding system, then the whole information element is ordered by that information encoding system too:

(A32) $InformationElement(x) \wedge DOLCE : part(x, z, t) \wedge orderedBy(z, y, t) \rightarrow$
$orderedBy(x, y, t)$

If an information element is ordered by an information encoding system, then either it is entirely ordered by such a system or it has a proper part that it is:

(A33) $orderedBy(x, y, t) \rightarrow (entirelyOrderedBy(x, y, t) \vee$
$(\exists z)(DOLCE : properPart(x, z, t) \wedge entirelyOrderedBy(z, y, t))$

The property $entirelyOrderedBy(x, y, t)$ is stronger than $orderedBy(x, y, t)$:

(A34) $entirelyOrderedBy(x, y, t) \rightarrow orderedBy(x, y, t)$

If an information encoding system entirely orders an information element, then it entirely orders also any proper part of that information element:

(A35) $entirelyOrderedBy(x, y, t) \rightarrow$
$(\forall z)(DOLCE : properPart(x, z, t) \rightarrow entirelyOrderedBy(z, y, t))$

Conversely, if an information encoding system orders each proper part of a non-atomic information element, then it entirely orders also the whole information element:

(A36) $InformationElement(x) \wedge DOLCE : properPart(x, z1, t) \wedge$
$(\forall z)(DOLCE : properPart(x, z, t) \rightarrow entirelyOrderedBy(z, y, t)) \rightarrow$
$entirelyOrderedBy(x, y, t)$

The semantics of an information element is given (at least in part) by the particulars that the information element talks about and by the descriptions that it expresses. We refine the predicate $OIO : about(x, y, t)$ by stating that an information element is about all the particulars that its parts are about:

(A37) $InformationElement(x) \wedge DOLCE : part(x, z, t) \wedge OIO : about(z, y, t) \rightarrow$
$OIO : about(x, y, t)$

We do not specify the converse axiom stating that if a non atomic information element is about a particular p, then it necessarily has a proper part which is about p. In other words, we adhere to a holistic view of aboutness and we admit the possibility that two or more information elements together may be about some particular even though none of them, individually, is about it. For instance, a database record may be about an individual association between an employee and its department, while no field in the record is about this association.

In our domain we need to distinguish two more specific kinds of aboutness, namely the *identification* and the *representation*. Indeed, in an enterprise information system some information elements can be used to *identify* (other) particulars or also to *represent* those particulars within the information system. For instance, an order code is used to identify an order relationship, while an order record (see below) is used to represent an order within the enterprise information system. In Oberle *et al.* [2006], the authors introduce the property $identifies$ as a specialization of $OIO : about$ to express that a (abstract) data in a computational system "identifies something different from itself", as it is for "a user account in a Unix operating system which has a physical counterpart in the real world". We also introduce the predicate $identifies(x, y, t)$, but we do not restrict it to the case of data within computational systems:

(A38) $identifies(x, y, t) \rightarrow InformationElement(x) \wedge DOLCE : Particular(y) \wedge DOLCE :$
$TimeInterval(t) \wedge OIO : about(x, y, t) \wedge x \neq y$

Differently from aboutness, for the sake of generality, we do not state, in general, that any information element identifies all the particulars that are identified by its parts.

As mentioned above, an information element can not only identify a particular, but it can also represent it within the enterprise information system. We thus introduce the predicate $represents(x, y, t)$ to specify the special case of identification where the information element x provides a characterization of the particular y complete enough to be considered as a sort of representation of y:

(A39) $represents(x, y, t) \rightarrow identifies(x, y, t)$

As regards the expression of descriptions, any information element expresses all the descriptions expressed by its parts:

(A40) $InformationElement(x) \wedge DOLCE : part(x, z, t) \wedge OIO : expresses(z, y, t) \rightarrow OIO : expresses(x, y, t)$

As with aboutness, we admit that two or more information elements together may express a description even though none of them, individually, expresses it.

Moreover, saying that an information element x expresses a description means stating that it expresses the *whole* description, and thus, if the expressed description has some proper parts that are descriptions, then x expresses also them:

(A41) $OIO : expresses(x, y, t) \wedge InformationElement(x) \rightarrow$
$(\forall z)(DnS : Description(z) \wedge DOLCE : properPart(y, z, t) \rightarrow OIO : expresses(x, z, t))$

Obviously, the converse is not true in general: an information element may express all the descriptive parts of a given description, without expressing that given description.

To complete the basic characterization of the information elements we need also to introduce a weak and a strong notion of equivalence between information elements. The weak notion of equivalence (expressed by the property $semanticallyEquivalent$) accounts for the semantic equivalence, but it does not require the equivalence of the information encoding systems that order the two information elements: For instance, two different reports may have the same content, even though one is written in English and the other is written in Italian. The strong notion of equivalence between information elements (expressed by the property $equivalentIE$), besides the semantic equivalence, requires also the equivalence of the ordering information encoding systems: For instance, sometimes an information may be duplicated and the two copies may evolve independently from each other; in those situations it may be convenient to regard the two copies as two different information elements since the beginning. At the time of duplication, however, the two information elements are not only semantically equivalent, but they are also ordered in the very same way.

Two information elements that are semantically equivalent have the same meaning, which implies that they are about, they identify and they represent, the same particulars and, moreover, they express the same descriptions:

(A42) $semanticallyEquivalent(x, y, t) \rightarrow InformationElement(x) \wedge$
$InformationElement(y) \wedge DOLCE : TimeInterval(t)$

(A43) $semanticallyEquivalent(x, y, t) \rightarrow$
$InformationElement(x) \wedge InformationElement(y) \wedge$
$(\forall z)(OIO : about(x, z, t) \leftrightarrow OIO : about(y, z, t)) \wedge$
$(\forall z)(identifies(x, z, t) \leftrightarrow identifies(y, z, t)) \wedge$
$(\forall z)(represents(x, z, t) \leftrightarrow represents(y, z, t)) \wedge$
$(\forall z)(OIO : expresses(x, z, t) \leftrightarrow OIO : expresses(y, z, t))$

Obviously, the semantic equivalence is a reflexive, symmetric and transitive relation between information elements:

(A44) $InformationElement(x) \rightarrow semanticallyEquivalent(x, x, t)$

(A45) $semanticallyEquivalent(x, y, t) \leftrightarrow semanticallyEquivalent(y, x, t)$

(A46) $semanticallyEquivalent(x, y, t) \wedge semanticallyEquivalent(y, z, t) \rightarrow$
$semanticallyEquivalent(x, z, t)$

The following axiom formalizes the strong notion of equivalence between two information elements x and y. Such a notion implies not only the semantic equivalence between the two information elements, but it also entails the equivalence between the encoding systems, in the sense that for each part of x, a corresponding part of y exists that is semantically equivalent to the first one and ordered in the very same way, and vice versa:

(A47) $equivalentIE(x, y, t) \rightarrow$
$semanticallyEquivalent(x, y, t) \wedge$
$(\forall x1, l)(DOLCE : part(x, x1, t) \wedge entirelyOrderedBy(x1, l, t) \rightarrow$
$(\exists y1)(DOLCE : part(y, y1, t) \wedge semanticallyEquivalent(x1, y1, t) \wedge$
$entirelyOrderedBy(y1, l, t))) \wedge$
$(\forall y1, l)(DOLCE : part(y, y1, t) \wedge entirelyOrderedBy(y1, l, t) \rightarrow$
$(\exists x1)(DOLCE : part(x, x1, t) \wedge semanticallyEquivalent(y1, x1, t) \wedge$
$entirelyOrderedBy(x1, l, t)))$

Like the semantic equivalence, the strong equivalence between information elements is a reflexive, symmetric and transitive relation:

(A48) $InformationElement(x) \rightarrow equivalentIE(x, x, t)$

(A49) $equivalentIE(x, y, t) \leftrightarrow equivalentIE(y, x, t)$

(A50) $equivalentIE(x, y, t) \wedge equivalentIE(y, z, t) \rightarrow$
$equivalentIE(x, z, t)$

Given the importance of the computer-supported knowledge management in the CRM domain, among the information elements O-CREAM-v2 distinguishes those that are ordered only by binary computer languages: These are called *digital information elements*. We can not report here their formal characterization, however it is worth mentioning that their physical realizations are always based either on a piece of hardware or on an electromagnetic wave. Moreover, we consider a *file* as a physical realization of a digital information element depending on some mass storage device.

Sometimes the information elements are used to *encode* other information elements; i.e., an information element can provide a representation for another information element such that the latter can be derived from the former. This is very frequent for the digital information elements. For instance, a JPEG information element encodes an image such that an image editor can reconstruct the encoded image from the encoding element. O-CREAM-v2 puts at disposal the predicate $encodes(x, y, t)$ to express that an information element x encodes another information element y (different from x) during time t.

Thus in the O-CREAM-v2 framework it it possible to state, for instance, that a JPEG file is the physical realization of an information element ordered by the JPEG standard; such element *encodes* an image; the encoded image is a specific kind of information element, it is ordered by some visual language and it can be physically realized by its appearance on a display (a special type of $OIO : InformationRealization$).[10]

4.2.2. (Business) Knowledge: Lower Core

The basic characterization of information elements has been exploited to specify several types of information elements that we encounter in the CRM domain: Figure 4 depicts some of them. Here we illustrate the approach by formally characterizing some notions relevant to the master data and to the reports.

An enterprise adopting CRM strategies usually needs to keep track of (almost) all the business relationships it is involved in. Not only the business relationships coming from the classical transactions (such as those relevant to the orders, the sales, the payments, etc.), but also those coming from more informal activities (such as those relevant to the communications, the appointments, the complaints, etc.), have to be represented within the enterprise information system. Therefore, the master data files that store the information elements representing the business relationships are an important part of the enterprise business knowledge.

[10]COMM (Core Ontology for Multimedia Annotation) Arndt *et al.* [2009] is an ontology for the "semantic representation of media objects", which extends DOLCE, DnS and OIO. It puts at disposal a rich conceptualization that encompasses the notions of digital data (e.g. image data, audio data, etc.) and media (e.g. image, etc.). Thus, despite its different scope and target domain, COMM has a little conceptual overlap with O-CREAM-v2. However, COMM does not explicitly account for the notion of *encoding* and it suggests a different representation pattern for describing the state of affairs similar to the example above. According to that pattern, for instance, an actual image can be the physical realization of a JPEG digital data and not of the information element that the digital data encodes, as it is in O-CREAM-v2, instead.

We formalize the notion of business relationship record by defining it as an information element representing a business relationship:

(D3) $BusinessRelationshipRecord(x) \equiv InformationElement(x) \wedge$
$(\exists y)(BusinessRelationship(y) \wedge$
$(\forall t)(DOLCE : TimeInterval(t) \wedge DOLCE : presentAt(x,t) \rightarrow represents(x,y,t)))$

It is worth noting that, given the definition above, during its whole life span, a business relationship record represents the same business relationship.

Each business relationship record always is a proper part of a business relationship master data:

(A51) $BusinessRelationshipRecord(x) \wedge DOLCE : presentAt(x,t) \rightarrow$
$(\exists y)(BusinessRelationshipMasterData(y) \wedge DOLCE : properPart(y,x,t))$

A business relationship master data is an information element which, as a whole, represents a (possibly empty) set of business relationships, all involving the same enterprise. This does not entail that a business relationship master data, as a whole, also represents the business relationships belonging to the represented set. However, we require that for each business relationship belonging to the represented set, the business relationship master data contains a record that represents the relationship. Conversely, each business relationship record in the master data always represents a member of the represented set of relationships. Formally, the notion of $BusinessRelationshipMasterData$ is characterized as follows:[11]

(A52) $BusinessRelationshipMasterData(x) \rightarrow$
$InformationElement(x) \wedge (\forall t)(DOLCE : TimeInterval(t) \wedge DOLCE : presentAt(x,t) \rightarrow$
$(\exists y)(Set(y) \wedge$
$(\forall z)(hasMember(y,z) \rightarrow BusinessRelationship(z)) \wedge$
$(\forall z1, z2)(hasMember(y,z1) \wedge hasMember(y,z2) \rightarrow$
$(\exists e)(Enterprise(e) \wedge hasRelationshipElement(z1,e) \wedge hasRelationshipElement(z2,e))) \wedge$
$extensionallyRepresents(x,y,t) \wedge$
$(\forall z)(hasMember(y,z) \rightarrow$
$(\exists brr)(BusinessRelationshipRecord(brr) \wedge DOLCE : properPart(x,brr,t) \wedge$
$represents(brr,z,t))) \wedge$
$(\forall brr)(BusinessRelationshipRecord(brr) \wedge DOLCE : properPart(x,brr,t) \rightarrow$
$(\exists z)(hasMember(y,z) \wedge represents(brr,z,t)))))$

It is worth noting that, given the axiom above, the set of relationships represented by a business relationship master data may change during the master data life span (i.e., the master data may represent different sets at different times).

For each business relationship type present in O-CREAM-v2 ($ContactRelationship$, $OrderRelationship$, $SaleRelationship$, etc.), in the ontology there are both a corresponding specific kind of business relationship record and a corresponding specific kind of business relationship master data. For instance, a $ContactRelationship$ can be represented by a $ContactRecord$, which is characterized as follows:

(D4) $ContactRecord(x) \equiv InformationElement(x) \wedge$
$(\exists y)(ContactRelationship(y) \wedge$
$(\forall t)(DOLCE : TimeInterval(t) \wedge DOLCE : presentAt(x,t) \rightarrow represents(x,y,t)))$

[11] O-CREAM-v2 provides a light charaterization for the notion of set in its Miscellaneous module (Section 4.5). Such a characterization, besides the *Set* category, also contains the properties $hasMember(x,y)$ and $extensionallyRepresents(x,y,t)$: all of them are used in Axioms A52 and A54. The former expresses the membership of the element y to the set x. The latter formalizes a special way in which an information element x can represent a set y during t (the general *representation* relation between an information element and a particular - i.e. $represents(x,y,t)$ - is introduced in Section 4.2.1). Intuitively: an information element *extensionally* represents a set if, besides representing that set, for each element of the set it also has both a proper part that represents that element and a proper part that represents the membership of the element to the set.

Given that a contact relationship is a particular kind of business relationship (Axiom $A13$), from the definition above and Definition $D3$, we derive that:

(T2) $ContactRecord(x) \rightarrow BusinessRelationshipRecord(x)$

Each contact record always is a proper part of a contact master data:

(A53) $ContactRecord(x) \wedge DOLCE : presentAt(x,t) \rightarrow$
$(\exists y)(ContactMasterData(y) \wedge DOLCE : properPart(y,x,t))$

A contact master data is a particular type of business relationship master data. As a whole, a contact master data, represents a (possibly empty) set of contact relationships of the same enterprise. Each contact record within a contact master data represents an individual contact relationship. The following axiom provides a formal characterization for the notion:

(A54) $ContactMasterData(x) \rightarrow$
$BusinessRelationshipMasterData(x) \wedge (\forall t)(DOLCE : TimeInterval(t) \wedge DOLCE : presentAt(x,t) \rightarrow$
$(\exists y)(Set(y) \wedge$
$(\forall z)(hasMember(y,z) \rightarrow ContactRelationship(z)) \wedge$
$(\forall z1,z2,w1,w2)(hasMember(y,z1) \wedge hasMember(y,z2) \wedge$
$hasContactHolderEnterprise(z1,w1) \wedge hasContactHolderEnterprise(z2,w2) \rightarrow$
$w1 = w2) \wedge extensionallyRepresents(x,y,t) \wedge$
$(\forall z)(hasMember(y,z) \rightarrow$
$(\exists cr)(ContacRecord(cr) \wedge DOLCE : properPart(x,cr,t) \wedge$
$represents(cr,z,t))) \wedge$
$(\forall cr)((ContacRecord(cr) \wedge DOLCE : properPart(x,cr,t) \rightarrow$
$(\exists z)(hasMember(y,z) \wedge represents(cr,z,t)))))$

Of course, everything that holds for the business relationship master data and for the business relationship records holds also for the particular cases of the contact master data and the contact records, respectively. In particular, an enterprise may acquire and lose contacts during its activity, therefore the same contact master data may represent different sets of contact relationships at different times. Differently, the same contact record may change in time (e.g., some parts can be added, deleted, modified, etc.), but it always represents the same contact relationship during its whole life span.

Besides the master data, the CRM processes involve several other kinds of information elements: they manage many documents, such as commercial documents (invoices, documents specifying orders, transport documents, etc.), promotional documents, reports, etc. In particular, the enterprises that adopt CRM strategies need to access the right knowledge at the right moment, therefore they pay much attention to the data analysis and the reporting activities. In many cases, the reports are relevant to business relationships (sale, payment, complaint relationships, etc.). As examples, in the following we provide a formal characterization for the reports on business relationship and for the particular cases of reports on sales.

A report on sales is simply an information element talking about business relationships or about sets of business relationships:

(A55) $ReportOnBusinessRelationships(x) \rightarrow InformationElement(x) \wedge$
$(\forall t)(DOLCE : presentAt(x,t) \rightarrow (\exists y)(about(x,y,t) \wedge (BusinessRelationship(y) \vee (Set(y) \wedge$
$(\forall z)(hasMember(x,z) \rightarrow BusinessRelationship(z))))))$

A report on sales is a particular report on business relationships that talks about sales relationships or about sets of sales relationships:

(A56) $ReportOnSales(x) \rightarrow ReportOnBusinessRelationships(x) \wedge$
$(\forall t)(DOLCE : presentAt(x,t) \rightarrow (\exists y)(about(x,y,t) \wedge (SaleRelationship(y) \vee (Set(y) \wedge$
$(\forall z)(hasMember(x,z) \rightarrow SaleRelationship(z))))))$

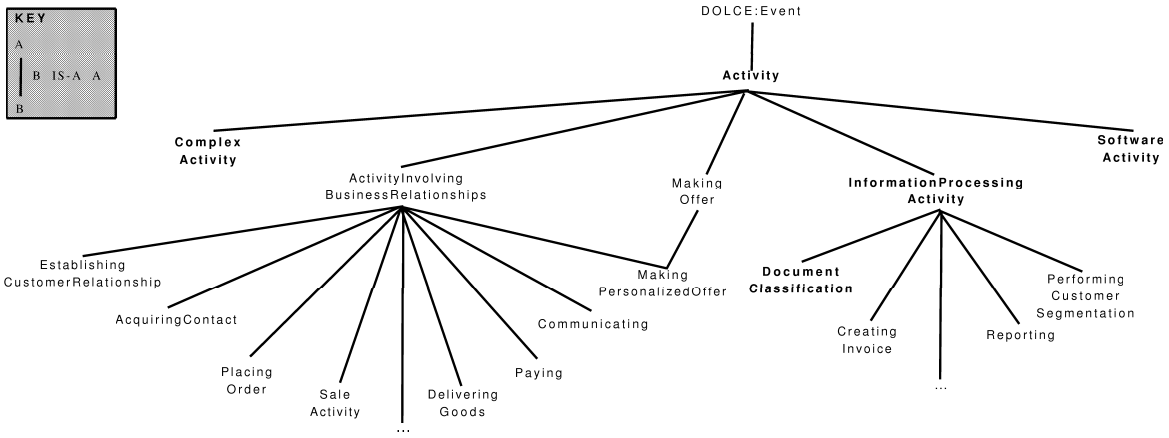

Fig. 6. A fragment of O-CREAM-v2 Activity Taxonomy

It is worth noting that a report on business relationships may talk about individual business relationships, by providing, for instance, some information on each of them; however, it can also provide some collective information on relationships (e.g., the means of the paid prices for different kinds of products in the sales in the last three months). Such a collective information refers to sets of business relationships and not to the individual relationships: This is why we explicitly mention both the individual relationships and the sets of such relationships in the two axioms above.

4.3. (Business) Activities

In the considered domain, the notion of *activity* plays a central role. This is a rather broad concept that encompasses, among others: the core business activities in the CRM domain (e.g., acquiring a new contact, selling a product, making an offer, communicating with a customer, etc.); the activities related to information management (e.g., reporting, handling documents, performing a statistical analysis on sales, clustering customers into different groups sharing similar characteristics, etc.); the specific activities supporting the main processes (e.g., creating a bill, printing an invoice, changing the format of an electronic document, etc.) and, given that software applications are of paramount importance in CRM, all the activities resulting from the execution of any piece of software.

Figure 6 depicts a fragment of the O-CREAM-v2 taxonomy of activities (the names of the concepts belonging to the O-CREAM-v2 upper core are in bold).

Despite the differences between activities of different kinds, they all share the same set of characteristics that are captured by the general $Activity$ concept: Each activity always has a starting time and can have an ending one; it can receive some particulars as inputs and involve some particulars as outputs; it is always executed by someone or something; it can exploit some resources (either by using or consuming them) and it may be performed according to some specified methods. Figure 7 reports the most relevant properties that characterize O-CREAM-v2 activities (as in the figures above, those belonging to the upper core are written in bold, the dashed lines specifies a sub-property/super-property relationship, while the dotted lines represent an entailment relationship between two properties different from a sub-property/super property relationship).

In the present section we describe the main characteristics of the O-CREAM-v2 activities. We start by formally characterizing the basic $Activity$ and $ComplexActivity$ concepts. In particular, since any activity can modify the state of the world by involving one or more particulars, we specify several different ways in which this involvement can take place. Moreover, given their importance in the CRM do-

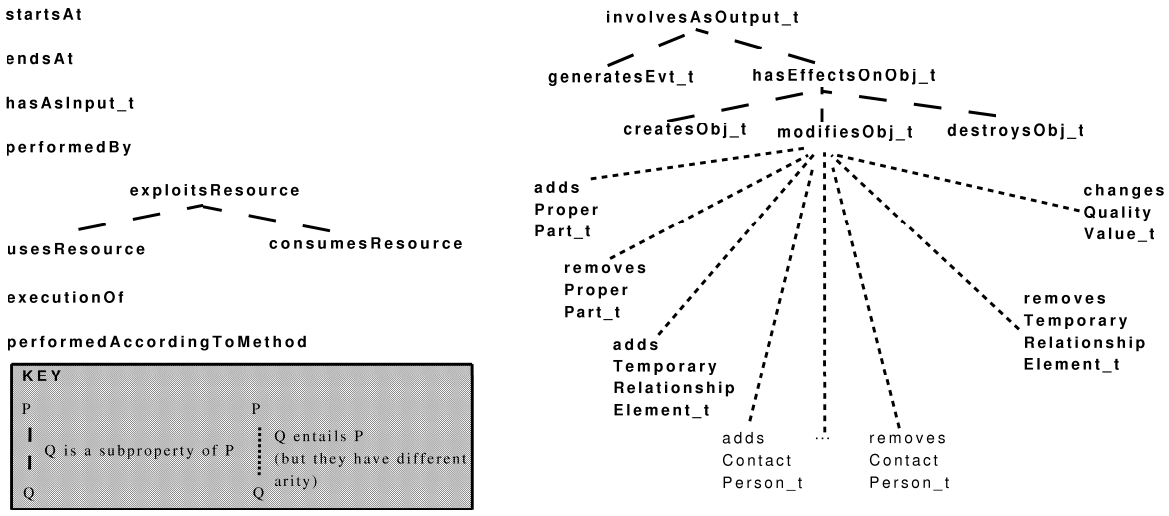

Fig. 7. Most relevant properties characterizing O-CREAM-v2 activities

main, we also provide a characterization both for the information processing and the software activities. Furthermore, the domain analysis showed that in CRM there is a fundamental distinction between the relationships and the activities that create (and, more generally, involve) them. This means, in particular, that a business relationship and the activity that has created it are two distinct particulars: For instance, a sale activity and the new sale relationship that it establishes are two different elements of the world. This section also introduce the notion of *activity involving business relationships* and provide some examples for it.[12]

4.3.1. (Business) Activities: Upper Core

First of all, activities are special kinds of events:

(A57) $Activity(x) \rightarrow DOLCE : Event(x)$[13]

Each activity has always one and only one starting time:

(A58) $startsAt(x,t) \rightarrow DOLCE : Event(x) \wedge TimeInstant(t)$[14]

(A59) $Activity(x) \rightarrow (\exists t)(startsAt(x,t))$

(A60) $Activity(x) \wedge startsAt(x,t1) \wedge startsAt(x,t2) \rightarrow t1 = t2$

In a similar way, an activity can have one (and no more than one) ending time:[15]

(A61) $endsAt(x,t) \rightarrow DOLCE : Event(x) \wedge TimeInstant(t)$

(A62) $Activity(x) \wedge endsAt(x,t) \wedge endsAt(x,t2) \rightarrow t1 = t2$

Obviously, the starting time must precede the ending one:

[12] In the considered domain, temporary relations between the activities and other particulars (executors, resources, etc.) are usually not needed. This is why, for the sake of simplicity, whenever it is possible, we omit the temporal parameter in the modeled relations. However, in O-CREAM-v2, we do use temporary relations in those cases where they are needed in order to better capture their meaning or to better represent the relationships among them.

[13] It is worth noting that since $Activity$ is a subclass of $DOLCE : Event$, then the $Activity$ class denotes the set of "actual" activities, which should not be confused with their descriptions or conceptualizations. As stated in Section 3.2, DnS Gangemi and Mika [2003]; Gangemi *et al.* [2005] puts at disposal the notion of *course*, represented by the $DnS : Course$ class, whose instances are the descriptive counterparts of events. Within that framework, an activity description can be brought into the domain of discourse by modeling it as a $DnS : Description$ that $DnS : defines$ a $DnS : Course$, which $DnS : sequences$ the described actual activity. In this chapter, we focus on the characterization of actual activities.

[14] Time istants are atomic time intervals, i.e.: $TimeInstant(t) \equiv DOLCE : TimeInterval(t) \wedge DOLCE : Atom(t)$.

[15] Since we need to be able to talk also about running activities and given that, for the sake of generality, in this core ontology we admit never-ending activities, we do not specify the ending time as mandatory for activities.

(A63) $Activity(x) \wedge startsAt(x, t1) \wedge endsAt(x, t2) \rightarrow t1 < t2$[16]

An activity can not be present before its starting time, while it is present at the time it starts (any event extends in time and, thus it can only be partially present at any time instant) and it remains present until its (possible) ending time; it ceases to be present at its ending time and it is no more present afterwards:

(A64) $Activity(x) \wedge startsAt(x, t) \rightarrow (\forall t1)(t1 < t \rightarrow \neg DOLCE : presentAt(x, t1)) \wedge DOLCE : presentAt(x, t)$

(A65) $Activity(x) \wedge startsAt(x, t1) \wedge endsAt(x, t2) \rightarrow$
$(\forall t3)(t1 \leq t3 < t2 \rightarrow DOLCE : presentAt(x, t3)) \wedge$
$(\forall t4)(t2 \leq t4 \rightarrow \neg DOLCE : presentAt(x, t4))$

Each object can participate in an activity only from the starting time of the activity up to its (possible) ending time:

(A66) $Activity(x) \wedge startsAt(x, t) \wedge DOLCE : participant(x, y, t1) \rightarrow$
$t \leq t1 \wedge (\forall t2)(endsAt(x, t2) \rightarrow t1 \leq t2)$

There may be complex activities that have subactivities as their parts:

(D5) $ComplexActivity(x) \equiv Activity(x) \wedge (\exists y)(Activity(y) \wedge DOLCE : properPart(x, y))$

If a complex activity starts at time t, then there must be at least one subactivity that starts at t and no subactivity can start before t:

(A67) $ComplexActivity(x) \wedge startsAt(x, t) \rightarrow$
$(\exists y)(Activity(y) \wedge DOLCE : properPart(x, y) \wedge startsAt(y, t)) \wedge$
$(\forall y, t1)(Activity(y) \wedge DOLCE : properPart(x, y) \wedge startsAt(y, t1) \rightarrow t \leq t1)$

Similarly, if a complex activity ends at a time t, then there must be at least one subactivity that ends at t and no subactivity can end after t:

(A68) $ComplexActivity(x) \wedge endsAt(x, t) \rightarrow$
$(\exists y)(Activity(y) \wedge DOLCE : properPart(x, y) \wedge endsAt(y, t)) \wedge$
$(\forall y)(Activity(y) \wedge DOLCE : properPart(x, y) \rightarrow (\exists t1)(endsAt(y, t1) \wedge t1 \leq t))$

If each subactivity of a complex activity has an ending time, then the complex activity itself has an ending time too:

(A69) $ComplexActivity(x) \wedge$
$(\forall y)(Activity(y) \wedge DOLCE : properPart(x, y) \rightarrow (\exists t)(endsAt(y, t))) \rightarrow$
$(\exists t1)(endsAt(x, t1))$

As stated above, an activity can receive as inputs, or involve as outputs, some particulars. Very often, the activities receive as inputs only objects, such as amounts of iron or some customer data. Similarly, in many cases, the outputs of activities involve only objects: For instance, the result of an activity can be a new technological artifact (such as a car) or an information element (such as a report on customers), etc. However, in the considered domain, also events can be inputs for (or outputs of) activities. For instance, a speech-to-text system actually receives as input a speech, which is an event; conversely, a text-to-speech system generates a speech (i.e., an event). We thus admit both objects and events as inputs and outputs of activities.

We firstly consider the inputs, by characterizing the predicate $hasAsInput_t(x, y, t)$, whose intuitive meaning is that *at time t, y is available as input for the activity* x:

[16] Given two time instants $t1$ and $t2$, the predicate "$t1 < t2$" means that $t1$ *precedes* $t2$. We will also use the shortcuts "$t1 \leq t2$", instead of "$(t1 < t2) \vee (t1 = t2)$", which means that $t1$ *does not follow* $t2$ and "$t1\ Op_1\ t2\ Op_2\ t3$", (where Op_1 and Op_2 can be either $<$ or $\leq$), instead of "$(t1\ Op_1\ t2) \wedge (t2\ Op_2\ t3)$".

(A70) $hasAsInput_t(x, y, t) \rightarrow Activity(x) \wedge (DOLCE : Object(y) \vee DOLCE : Event(y)) \wedge TimeInstant(t) \wedge DOLCE : presentAt(x, t) \wedge DOLCE : presentAt(y, t)$[17]

Obviously, any element y can be available as input to an activity x only during the activity itself:

(A71) $hasAsInput_t(x, y, t) \wedge startsAt(x, t1) \rightarrow t1 \leq t \wedge (\forall t2)(endsAt(x, t2) \rightarrow t \leq t2)$

The inputs to an activity that are objects participate in the activity (at least for all the times instants that they are available as inputs):

(A72) $hasAsInput_t(x, y, t) \wedge DOLCE : Object(y) \rightarrow DOLCE : participant(x, y, t)$

Having as input an object implies having as inputs also all its (possible) proper parts:

(A73) $hasAsInput_t(x, y, t) \wedge DOLCE : Object(y) \wedge DOLCE : properPart(y, z, t) \rightarrow hasAsInput_t(x, z, t)$

It is worth noting that if y is an event no axiom analogous to the latter holds. Indeed, events "extend in time" Masolo *et al.* [2003] and, thus, we can not say that all the temporal parts of y are available as inputs to x, at any time t when the predicate $hasAsInput_t(x, y, t)$ is true.

An activity may also involve some particulars as outputs, namely, it can generate some events and it can produce some effects on objects, by creating, destroying or modifying them. In order to account for the main possibilities, we introduce in the upper core several temporary relations; the principal ones are (see Figure 7): $involvesAsOutput_t$, $generatesEvt_t$, $hasEffectsOnObj_t$, $createsObj_t$, $destroysObj_t$, $modifiesObj_t$, $addsProperPart_t$, $removesProperPart_t$, $addsTemporaryRelationshipElement_t$, $removesTemporaryRelationshipElement_t$ and $changesQualityValue_t$. The lower core of the ongology accounts for other kinds of activity outputs, more specific to the CRM (see Section 4.3.2).

The relation $involvesAsOutput_t$ is the most general one holding between an activity and its outputs and it captures both the generation of events, and the production of effects on objects. Intuitively, $involvesAsOutput_t(x, y, t)$ means that *at time t, the activity x somehow involves y as its output* (we will distinguish in the following some relevant ways in which an activity can involve a particular as its output):

(A74) $involvesAsOutput_t(x, y, t) \rightarrow Activity(x) \wedge (DOLCE : Object(y) \vee DOLCE : Event(y)) \wedge TimeInstant(t) \wedge DOLCE : presentAt(x, t) \wedge DOLCE : presentAt(y, t)$

As stated for inputs, any element y can be involved as output by an activity x only during the activity itself:

(A75) $involvesAsOutput_t(x, y, t) \wedge startsAt(x, t1) \rightarrow t1 \leq t \wedge (\forall t2)(endsAt(x, t2) \rightarrow t \leq t2)$

In general, it is possible that the same activity involves as output the same element several times. This means, in particular, that the involvement as output may extend for an interval of time (and it is not necessarily istantaneous, even though it could be).

Moreover, similarly to inputs, any object involved as output by an activity participates in the activity (at least for all the times it is involved as output):

(A76) $involvesAsOutput_t(x, y, t) \wedge DOLCE : Object(y) \rightarrow DOLCE : participant(x, y, t)$

The generation of an event is a specific case of output. We thus introduce the predicate $generatesEvt_t(x, y, t)$, whose intuitive meaning is: *at time t, the activity x is generating the event y.*

(A77) $generatesEvt_t(x, y, t) \rightarrow involvesAsOutput_t(x, y, t) \wedge DOLCE : Event(y)$

A generated event is not present when the generating activity is not generating it:

[17] For each relation $R_t(\overline{x}, t)$ (where $\overline{x}$ is a list of non-temporal parameters), $R(\overline{x}) \equiv (\exists t)R_t(\overline{x}, t)$ is the corresponding non temporary one. For instance, $hasAsInput(x, y) \equiv (\exists t)(hasAsInput_t(x, y, t))$.

(A78) $(\exists t1)(generatesEvt_t(x,y,t1)) \rightarrow$
$(\forall t)(\neg generatesEvt_t(x,y,t) \rightarrow \neg DOLCE : presentAt(y,t))$

Since an event extends in time, its generation by an activity extends in time too, but there must always be a specific moment in which the activity has started generating it. Moreover, from that moment until any time t in which the predicate $generatesEvt_t(x,y,t)$ is true, the generating activity has never stopped generating it (and, consequently, the generated event has never ceased to be present):

(A79) $generatesEvt_t(x,y,t) \rightarrow (\exists t1)(TimeInstant(t1) \wedge t1 \leq t \wedge generatesEvt_t(x,y,t1) \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t2 < t1 \rightarrow \neg generatesEvt_t(x,y,t2)) \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 \leq t \rightarrow generatesEvt_t(x,y,t2)))$

The predicate $hasEffectsOnObj_t(x,y,t)$ captures the relation between an activity and the objects which that activity produces effects on. Its intuitive meaning is that *at time t, the activity x produces some effects on the object y*. In general, the same activity can have effects on the same object many times (i.e. at different times). This relation is formalized by the following axioms and definitions:

(A80) $hasEffectsOnObj_t(x,y,t) \rightarrow involvesAsOutput_t(x,y,t) \wedge DOLCE : Object(y)$

An activity can produce effects on an object in three different ways, namely by creating, destroying or modifying it:

(D6) $hasEffectsOnObj_t(x,y,t) \equiv createsObj_t(x,y,t) \vee destroysObj_t(x,y,t) \vee modifiesObj_t(x,y,t)$

As regards the creation, the intended meaning of $createsObj_t(x,y,t)$ is that *at time t, the activity x creates the object y*. In O-CREAM-v2, creating an object means making it to become present (in DOLCE sense) and the creation time t specifies the very moment in which the created object y begins to be present, after an interval of time (possibly infinite) in which it was not present. This means, in particular, that only non-present elements can be created. However, for the sake of generality, we admit that the same element can be created more than once (i.e. in general, an object can alternate periods of non-presence to periods of presence and it can be brought from non-presence into presence by an activity that creates it).

The following axiom formally characterizes the notion of creation in our domain (it should be noted that, in principle, an element y created by an activity x can be destroyed by any activity z - possibly other than x - and this can happen also before x has ended, therefore we can not require that a created element is present at the time when the creating activity ends):

(A81) $createsObj_t(x,y,t) \rightarrow (\exists t1)(TimeInstant(t1) \wedge t1 < t \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow \neg DOLCE : presentAt(y,t2)))$

Moreover, it should be clear that from Definition $D6$ and Axioms $A74$ and $A80$, we derive

(T3) $createsObj_t(x,y,t) \rightarrow DOLCE : presentAt(y,t)$.

It is worth pointing out that creating an element does not necessarily mean creating all its (possible) proper parts: A new artifact, for instance, can be created by assembling a set of existing components.

Besides creating it, an activity can also destroy an existing particular. We specify the destruction of an object by means of the predicate $destroysObj_t(x,y,t)$, which means, informally: *at time t, the activity x destroys the object y*. The destruction is considered the opposite of the creation. Therefore, destroying an object means making it to cease to be present (in DOLCE sense) and the destruction time t specifies the very last moment in which the destroyed object is still present, immediately before a (possibly infinite) time interval in which the destroyed element is not present. As for creation, for the sake of generality, we admit that the same element can be destroyed more than once. The following axiom formally characterizes the notion of destruction in our domain (again, we note that from the definitions and axioms above, we can derive $destroysObj_t(x,y,t) \rightarrow DOLCE : presentAt(y,t)$):

(A82) $destroysObj_t(x,y,t) \rightarrow$
$(\exists t1)(TimeInstant(t1) \wedge t < t1 \wedge (\forall t2)(TimeInstant(t2) \wedge t < t2 \leq t1 \rightarrow \neg DOLCE :$
$presentAt(y,t2)))$

Furthermore, destroying an element does not necessarily mean destroying any of its (possible) proper parts: An assembled artifact, for instance, can be destroyed by simply disassembling it.

Differently from creating or destroying any element, an activity can have effects on particulars also by modifying them: $modifiesObj_t(x, y, t)$ informally means that *at time t, the activity x modifies the object y*. Modifying a proper part of any object means also modifying the object as a whole:

(A83) $modifiesObj_t(x, y, t) \wedge DOLCE : properPart(z, y, t) \rightarrow modifiesObj_t(x, z, t)$

We have identified several kinds of modifications that an activity may perform on particulars (see Figure 7). Some of them are rather general and perhaps useful beyond the CRM domain, thus they are captured by properties belonging to the upper core of the ontology. In the present section, we discuss only the most important of them. In particular we present: two predicates expressing mereological modifications (i.e., adding, and removing a proper part), two predicates relevant to the addition and the removal of relationship elements and one predicate accounting for the changing of a property value. Other modification types are very specific to the CRM domain and thus they are characterized in the lower core of O-CREAM-v2: In Section 4.3.2 we will discuss two of them (namely, $addsContactPerson_t$ and $removesContactPerson_t$).

As regards the addition of a new proper part, the predicate $addsProperPart_t(x, y, p, t)$ means, informally, that *at time t, the activity x adds to the object y a new proper part p*. The time t specifies the very moment in which the object p begins to be a proper part of y, after an interval of time (possibly infinite) in which it was not. Formally:

(A84) $addsProperPart_t(x, y, p, t) \rightarrow modifiesObj_t(x, y, t)$

(A85) $addsProperPart_t(x, y, p, t) \rightarrow DOLCE : Object(p) \wedge DOLCE : participant(x, p, t)$

(A86) $addsProperPart_t(x, y, p, t) \rightarrow$
$(\exists t1)(TimeInstant(t1) \wedge t1 < t \wedge (\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow \neg DOLCE : properPart(y, p, t2))) \wedge DOLCE : properPart(y, p, t)$

In principle, after x has added p as a proper part to y, p can be removed from y by any activity z - possibly other than x - and this can happen also before x has ended, therefore we can not require that the added proper part is still a proper part of y when the adding activity ends.

Conversely to the addition of a proper part, the predicate $removesProperPart_t(x, y, p, t)$ means, informally, that *at time t, the activity x removes from the object y its proper part p*. The time t specifies the very last moment in which p is still a proper part of y, immediately before a (possibly infinite) time interval in which p is no longer a proper part of y. Formally:

(A87) $removesProperPart_t(x, y, p, t) \rightarrow modifiesObj_t(x, y, t)$

(A88) $removesProperPart_t(x, y, p, t) \rightarrow$
$DOLCE : Object(p) \wedge DOLCE : participant(x, p, t)$

(A89) $removesProperPart_t(x, y, p, t) \rightarrow$
$DOLCE : properPart(y, p, t) \wedge (\exists t1)(TimeInstant(t1) \wedge t1 > t \wedge (\forall t2)(TimeInstant(t2) \wedge t < t2 \leq t1 \rightarrow \neg DOLCE : properPart(y, p, t2)))$

As a particular case, we make explicit that destroying a proper part p of y actually means removing it from y:

(A90) $destroysObj_t(x, p, t) \wedge DOLCE : properPart(y, p, t) \rightarrow removesProperPart_t(x, y, p, t)$

As we have seen (Section 4.1), the relationships can acquire and lose temporary relationship elements during their life. The activities can also add and remove temporary relationship elements to and from relationships. The properties $addsTemporaryRelationshipElement_t(x, y, e, t)$ and $removesTemporaryRelationshipElement_t(x, y, e, t)$ can be used to model these situations. In particular, the former means, informally, that *at time t, the activity x adds to the relationship y a new temporary*

relationship element e. The time t specifies the very moment in which the element e becomes a temporary element of y, after an interval of time (possibly infinite) in which it was not. Formally:

(A91) $addsTemporaryRelationshipElement_t(x, y, e, t) \rightarrow modifiesObj_t(x, y, t)$

(A92) $addsTemporaryRelationshipElement_t(x, y, e, t) \rightarrow$
$Relationship(y) \wedge DOLCE : Particular(e)$

(A93) $addsTemporaryRelationshipElement_t(x, y, e, t) \rightarrow$
$(\exists t1)(TimeInstant(t1) \wedge t1 < t \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow$
$\neg hasTemporaryRelationshipElement(y, e, t2))) \wedge$
$hasTemporaryRelationshipElement(y, e, t)$

Conversely, $removesTemporaryRelationshipElement_t(x, y, e, t)$ informally means that *at time* t*, the activity* x *removes from the relationship* y *its temporary element* e. The time t specifies the very last moment in which e is still a temporary relationship element of y, immediately before a (possibly infinite) time interval in which e is no longer a temporary relationship element of y. Formally:

(A94) $removesTemporaryRelationshipElement_t(x, y, e, t) \rightarrow modifiesObj_t(x, y, t)$

(A95) $removesTemporaryRelationshipElement_t(x, y, e, t) \rightarrow$
$Relationship(y) \wedge DOLCE : Particular(e)$

(A96) $removesTemporaryRelationshipElement_t(x, y, e, t) \rightarrow$
$hasTemporaryRelationshipElement_t(y, e, t) \wedge$
$(\exists t1)(TimeInstant(t1) \wedge t < t1 \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t < t2 \leq t1 \rightarrow$
$\neg hasTemporaryRelationshipElement_t(y, e, t2)))$

As particular case, destroying a temporary relationship element e of y always entails its removal from y:

(A97) $destroysObj_t(x, e, t) \wedge hasTemporaryRelationshipElement(y, e, t) \rightarrow$
$removesTemporaryRelationshipElement_t(x, y, e, t)$

Another way in which an activity can modify an object is by changing some of its quality values (the *quales*, in the DOLCE terminology). For instance, moving an object means changing the value of its space location (which can be modeled as a DOLCE quality). O-CREAM-v2 puts at disposal the predicate $changesQualityValue_t(x, y, q, t)$ for specifying the changing of quality values. The intended meaning of this predicate is that *at time* t*, the activity* x *changes the value taken by the individual quality* q *of* y. The time t represents the very moment in which q changes its value and takes the new one. The interpretations of the predicate are restricted as follows:

(A98) $changesQualityValue_t(x, y, q, t) \rightarrow modifiesObj_t(x, y, t)$

Since any modified particular y is a $DOLCE : Object$, it bears only physical or abstract qualities:

(A99) $changesQualityValue_t(x, y, q, t) \rightarrow$
$(DOLCE : PhysicalQuality(q) \vee DOLCE : AbstractQuality(q)) \wedge DOLCE : hasQuality(y, q)$

(A100) $changesQualityValue_t(x, y, q, t) \rightarrow$
$(\exists t1, qv1, qv2)(TimeInstant(t1) \wedge$
$((DOLCE : PhysicalRegion(qv1) \wedge DOLCE : PhysicalRegion(qv2)) \vee$
$(DOLCE : AbstractRegion(qv1) \wedge DOLCE : AbstractRegion(qv2))) \wedge$
$t1 < t \wedge (\forall t2)(TimeInstantl(t2) \wedge t1 \leq t2 < t \rightarrow DOLCE : qLocation(q, qv1, t2)) \wedge DOLCE :$
$qLocation(q, qv2, t) \wedge qv1 \neq qv2)$

Besides the creation, destruction, and modification of objects or events, accounted for by the relations described so far, each activity is always performed by someone or something. In our domain, only DOLCE objects (e.g., human persons, enterprises, computers, etc.) can perform activities:

(A101) $performedBy(x, y) \rightarrow Activity(x) \wedge DOLCE : Object(y)$

(A102) $Activity(x) \rightarrow (\exists y)(performedBy(x, y))$

Any object participates for at least one time instant in the activity that it performs, but it is not required that an executor participates in all parts of the activity (in particular, an activity may have different executors at different moments):

(A103) $performedBy(x, y) \wedge startsAt(x, t1) \rightarrow$
$(\exists t)(TimeInstant(t) \wedge t1 \leq t \wedge (\forall t2)(endsAt(x, t2) \rightarrow t \leq t2) \wedge DOLCE : participant(x, y, t))$

Any object performing a complex activity must always execute at least one subactivity (the converse is not stated, since we admit that any object performing a subactivity might not be considered also as an object carrying out the whole complex activity).

(A104) $ComplexActivity(x) \wedge performedBy(x, y) \rightarrow$
$(\exists z)(Activity(z) \wedge DOLCE : properPart(x, z) \wedge performedBy(z, y))$

It is worth pointing out that we can not state axioms similar to the previous one for either of the relation $hasAsInput_t$ or $involvesAsOutput_t$, etc. (introduced above). In fact, the analysis of the domain did not allow us to draw similar conclusions for these relations: For instance, an output of a business process can not always be considered as an ouput of one of its activities.

Besides having inputs and outputs, an activity can also exploit some resources, both by using them or by consuming them. O-CREAM-v2 offers the general predicate $exploitsResource(x, y)$ and the two more specific predicates $usesResource(x, y)$ and $consumesResource(x, y)$. We do not report here their formal characterization.

Furthermore, a notion of *method* is present in O-CREAM-v2 as a special kind of $DnS : Description$. It is represented by the class $Method$ and it accounts for for those descriptions that specify "how to do things". The property $performedAccordingToMethod(x, y)$ specifies that activity x is performed according to method y. For an activity x to be performed according to a method y we require that at least one part w of x is sequenced by a $DnS : Course\ z$ ($DnS : sequences(z, w, t)$) used by the method y ($DnS : uses(y, z)$).[18]

Given the central role that the knowledge management plays in the CRM domain, it is worth analyzing some features of those activities that process information elements, information physical realizations or descriptions (that are expressed by information elements). In particular, we provide a basic characterization for the information processing activities and then we make explicit the basic relationships between the information elements, their physical realizations and the expressed descriptions, when they are given as inputs to an activity or they are created by an activity. Furthermore, we also provide a formalization for $DocumentClassification$, a specific kind of information processing activity.

Each information processing activity is an activity that receives as input and/or involves as output some information elements:

(A105) $InformationProcessingActivity(x) \rightarrow Activity(x) \wedge$
$(\exists y, t)(InformationElement(y) \wedge TimeInstant(t) \wedge$
$(hasAsInput_t(x, y, t) \vee involvesAsOutput(x, y, t)))$

Although one could expect that the converse holds as well (i.e., that each activity that either receives as input or involves as output any information element always is an information processing activity), the outcomes of our analysis of the CRM domain advise against stating it in general. Indeed, almost all the

[18] Two notions of *method* are present both in http://www.loa-cnr.it/ontologies/ExtendedDnS.owl and http://ontologydesingpatterns.org/ont/dul/DUL.owl.

main activities in the considered domain actually access or produce some information, but classifying all of them as information processing activities would not reflect the point of view of the CRM practitioners. Similar considerations guided us in the characterization of many activity types for which we have specified only necessary conditions.

Whenever an information element is available as input to an activity, that activity has necessarily (received) as inputs also the physical realizations for all the parts of the information element. Indeed, there is no way to actually access an information element other than through the physical realizations of its parts. Formally:

(A106) $hasAsInput_t(x, y, t) \wedge InformationElement(y) \wedge DOLCE : part(y, z, t) \rightarrow$
$(\exists w, t1)(t1 \leq t \wedge OIO : InformationRealization(w) \wedge OIO : realizedBy(z, w, t1) \wedge$
$hasAsInput_t(x, w, t1))$

In the simplest case, for each information element y given as input to an activity x, a single physical realization w is given as input to x that entirely realizes y.

The converse statement of the above axiom seems not to hold in general. Let's consider, for instance, the activity of copying a file (which is a physical realization for a digital information element, as stated in Section 4.2.2): such an activity produces a new file, i.e. a new physical realization for a same digital information element; to perform that task, the activity surely needs to access the source file; however, it could be rather counterintuitive stating that it also accesses the information element actually realized by the source file. Therefore, in OCREAM-v2, we admit that an activity can receive as input an information realization without receiving as input also the corresponding realized information element.

Moreover, as regards the creation of information elements, we have to consider that every information element must always be somehow physically realized. Furthermore, different physical realizations can realize different parts of the same information element. Therefore, any activity x that creates an information element y must also create the physical realizations (in case they are objects) or generate them (in case they are events) for all the parts of y, unless they are already available as inputs to x:

(A107) $createsObj_t(x, y, t) \wedge InformationElement(y) \wedge DOLCE : part(y, z, t) \wedge$
$\neg(\exists w)(OIO : InformationRealization(w) \wedge OIO : realizedBy(z, w, t) \wedge$
$hasAsInput_t(x, w, t)) \rightarrow$
$(\exists w)(OIO : InformationRealization(w) \wedge OIO : realizedBy(z, w, t) \wedge$
$(\exists t1)(t1 \leq t \wedge createsObj_t(x, w, t1)) \vee generatesEvt_t(x, w, t))$

As in the case of input, the converse is not true in general, since an activity can create an information realization without necessarily creating the corresponding realized information elements. For instance, the activity of copying a file creates the destination file (i.e. the information physical realization), but it would be rather counterintuitive stating that it also creates the information elements realized by the file.

As far as the descriptions are concerned, if a description is available as input to an activity, then an information element expressing that description must be provided as input to the activity:

(A108) $hasAsInput_t(x, y, t) \wedge DnS : Description(y) \rightarrow$
$(\exists w, t1)(t1 \leq t \wedge InformationElement(w) \wedge OIO : expresses(w, y, t1) \wedge$
$hasAsInput_t(x, w, t1))$

Furthermore, an activity x that creates a description y must also create the information element that expresses the description, unless it is already available as input (in this last case, the creation of the description could have required, for instance, a modification to the expressing information element provided as input):

(A109) $createsObj_t(x, y, t) \wedge DnS : Description(y) \wedge$
$\neg(\exists w)(InformationElement(w) \wedge OIO : expresses(w, y, t) \wedge$
$hasAsInput_t(x, w, t)) \rightarrow$
$(\exists t1, w)(t1 \leq t \wedge InformationElement(w) \wedge createsObj_t(x, w, t1) \wedge OIO : expresses(w, y, t))$

Among the activities processing information elements, it is worth mentioning the document classification. This is a rather specific kind of information processing activity. However O-CREAM-v2 provides a characterization for it in its upper core, since, despite its specificity, such kind of activities play an important role in several knowledge management processes, also beyond the CRM domain.

As a basic characterization, we can describe a document classification activity x as an information processing activity that "classifies" at least one document doc in at least one set of documents ds. Namely: x is an information processing activity that receives as input either a document doc or an information element j about a document doc, and it creates an information element i representing the membership relationship m between the document doc and the container set ds. Formally (considering that $Document(d) \rightarrow InformationElement(d)$)[19]:

(A110) $DocumentClassification(x) \rightarrow InformationProcessingActivity(x) \wedge$
$(\exists doc, ds, m, i, t1, t)(Document(doc) \wedge Set(ds) \wedge MembershipRelationship(m) \wedge$
$InformationElement(i) \wedge TimeInstant(t1) \wedge TimeInstant(t) \wedge t1 \leq t \wedge$
$(hasAsInput_t(x, doc, t1) \vee (\exists j)(InformationElement(j) \wedge about(j, doc, t1) \wedge$
$hasAsInput_t(x, j, t1))) \wedge$
$(\forall d)(hasMember(ds, d) \rightarrow Document(d)) \wedge$
$hasContainerSet(m, ds) \wedge hasMemberElement(m, doc) \wedge represents(i, m, t) \wedge$
$createsObj_t(x, i, t)))$

Given the axiom above, it should be clear that in our context "classifying" means creating (at least) one information element that represents the membership between a document and a class of documents.

Moreover, we should point out that this basic characterization only specifies a mandatory input element (i.e., either a document or an information element about a document) and a mandatory output element (i.e., the information element representing the membership relationship). Nevertheless, a document classification activity can of course receive as input more than one document or some information about many documents and classify each of them in more than one class of documents. Furthermore, for the sake of generality, we can not state anything at this level on whether the information about the possible classes of documents is provided as input or it is created by the activity itself, nor we can formally characterize such kind of information.

Another important class of activities is that of software activities. Indeed, CRM heavily relies upon software applications that support its processes. In this domain, a software activity is an activity that results from the execution of some piece of software, as stated by the following axiom (the predicate $Software(x)$ states that x *is a piece of software* and it is defined in Section 4.4):

(D7) $SoftwareActivity(x) \equiv Activity(x) \wedge (\exists y)(Software(y) \wedge executionOf(x, y))$

A software activity can be performed only by computing systems, intended as hardware devices, such as PCs, PDAs, smart phones, etc. All of these elements are grouped in the $HwComputingSystem$ class, which is a class of Hardware Taxonomy, contained in the Miscellaneous module (Section 4.5).

(A111) $SoftwareActivity(x) \wedge performedBy(x, y) \rightarrow HwComputingSystem(y)$

Moreover, a piece of software participates in the activity resulting from its execution from the start of the activity until its (possible) end:

(A112) $executionOf(x, y) \rightarrow SoftwareActivity(x) \wedge Software(y)$

(A113) $executionOf(x, y) \wedge startsAt(x, t1) \rightarrow (\forall t)((TimeInstant(t) \wedge t1 \leq t \wedge (\forall t2)(endsAt(x, t2) \rightarrow t \leq t2)) \rightarrow DOLCE : participant(x, y, t))$

[19] The characterization of the notion of *set* in the Miscellaneous module (Section 4.5) includes a reification of the membership relationship between a set and an element of the set. In this way, such relationships belong to the domain of discourse, therefore it is possible to predicate on them by means of first-order predicates. The ontology vocabulary allows one to refer to any membership relationship x by means of the predicate $MembershipRelationship(x)$ and also to specify the elements of such a relationship, i.e. its container set y and each member z by means of the predicates $hasContainerSet(x, y)$ and $hasMemberElement(x, z)$, respectively. Axiom (A110) is an example of a possible usage of these ontology items.

4.3.2. (Business) Activities: Lower Core

The most representative activities accounted for in the lower core of O-CREAM-v2 are those involving business relationships and those performing specific information processing tasks.

The business relationships (between a company and its customers) are the main focus of the CRM, whose processes are primarily aimed at managing (i.e., creating, modifying, maintaining knowledge on, etc.) them. As stated above, O-CREAM-v2 assigns the business relationships an autonomous status w.r.t. the activities that create - and, in general, that involve - them. However, as we might expect, these kinds of activities play an important role too, in the considered domain.

An activity that involves some business relationship is simply an activity that either has as input or involves as output a business relationship:

(D8) $ActivityInvolvingBusinessRelationships(x) \equiv Activity(x) \wedge$
$(\exists y,t)(BusinessRelationship(y) \wedge TimeInstant(t) \wedge$
$(hasAsInput_t(x,y,t) \vee involvesAsOutput(x,y,t)))$

In certain cases the predicates $addsTemporaryRelationshipElement_t(x,y,e,t)$ and $removesTemporaryRelationshipElement_t(x,y,e,t)$ introduced above are not enough to express the specific modifications that some activities make to business relationships (and - more generally - to relationships). For instance, the basic characterization of $ContactRelationship$ introduces the predicates $hasContactPerson(x,y,t)$ and $hasCRQualifyingDescription(x,y,t)$ as sub-predicates of $hasTemporaryRelationshipElement(x,y,t)$ (see Section 4.1.2). Moreover, any ontology that extends O-CREAM-v2 might expand the vocabulary by introducing new sub-predicates of $hasTemporaryRelationshipElement(x,y,t)$ for characterizing $ContactRelationship$s. Therefore, there can be situations in which it is necessary to explicity specify whether an activity that adds a temporary relationship element to a contact relationship actually adds to the relationship a qualifying description, a contact person or something else. To easily cope with these situations, for each sub-predicate of $hasTemporaryRelationshipElement(x,y,t)$ occurring in its vocabulary, O-CREAM-v2 puts at disposal two corresponding predicates that express the addition and the removal of this specific kind of relationship-element temporary link. Here we formally illustrate this by considering only the predicate $hasContactPerson(x,y,t)$, but the same pattern has been followed for each sub-predicate of $hasTemporaryRelationshipElement(x,y,t)$ in the vocabulary of O-CREAM-v2.

The two predicates $addsContactPerson_t(x,y,p,t)$ and $removesContactPerson_t(x,y,p,t)$ enable us to model the addition and the removal of a contact person in a contact relationship. The former informally means that *at time t, the activity x adds to the relationship y a new contact person p* (and the time t specifies the very moment in which p becomes a new contact person in y), whereas the latter informally means that *at time t, the activity x removes p as a contact person in the relationship y*. The first predicate is restricted as follows:

(A114) $addsContactPerson_t(x,y,p,t) \rightarrow modifiesObj_t(x,y,t)$

(A115) $addsContactPerson_t(x,y,p,t) \rightarrow ContactRelationship(y) \wedge HumanPerson(p)$

(A116) $addsContactPerson_t(x,y,p,t) \rightarrow$
$(\exists t1)(TimeInstant(t1) \wedge t1 < t \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow$
$\neg hasContactPerson(y,p,t2))) \wedge hasContactPerson(y,p,t)$

Given that, in general, an element of a relationship may play more than one role within the relationship, adding a contact person p to a relationship y does not imply that p also becomes a new temporary relationship element; indeed, in general, nothing prevent p to already be a temporary relationship element in y (although not a contact person) immediately before and at the time when the activity x adds it as a contact person. However, if it is not the case (i.e., if p is not already a temporary element of y), then adding it as a contact person means also adding it as a new temporary element:

(A117) $addsContactPerson_t(x, y, p, t) \wedge (\exists t1)(TimeInstant(t1) \wedge t1 < t \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow$
$\neg hasTemporaryRelationshipElement(y, p, t2)) \rightarrow$
$addsTemporaryRelationshipElement_t(x, y, p, t))$

On the opposite, removing from y a temporary relationship element p which is (also) a contact person, means also removing it as a contact person:

(A118) $removesTemporaryRelationshipElement_t(x, y, e, t) \wedge hasContactPerson(y, e, t) \rightarrow$
$removesContactPerson_t(x, y, e, t)$

The predicate $removesContactPerson_t(x, y, p, t)$ is restricted in a way analogous to $addsContactPerson_t(x, y, p, t)$.

The ontology needs also to explicitly account for those activities that create business relationships. Therefore, each class of business relationships explicitly provided by the ontology (e.g., $CustomerRelationship$, $ContactRelationship$, $SaleRelationship$, etc.) is paired with the class of those activities that create that kind of relationships (a few of them are depicted in Figure 6). Therefore O-CREAM-v2 accounts for the notions of: $\underline{EstablishingCustomerRelationship}$ (which characterizes those activities that create new $CustomerRelationship$s); $\underline{AcquiringContact}$ (which accounts for those activities that create new $ContactRelationship$s); $\underline{SaleActivity}$ (which provides a formalization for those activities that create $SaleRelationship$s); etc.

In order to illustrate the approach, we here provide a formal characterization for the concept $AcquiringContact$. As stated above, acquiring contacts means creating (at least) a new contact relationship:

(D9) $AcquiringContact(x) \equiv ActivityInvolvingBusinessRelationships(x) \wedge$
$(\exists y, t)(ContactRelationship(y) \wedge TimeInstant(t) \wedge createsObj_t(x, y, t))$

Each activity x that creates a contact relationship y between an enterprise e and a contact c always entails a refinement of the customer relationship between e and c by the new created contact relationship, if such a customer relationship already exists (Axiom $A136$), otherwise it is the activity x itself that creates also the customer relationship (Axiom $A120$):[20]

(A119) $AcquiringContact(x) \wedge ContactRelationship(y) \wedge hasContactHolderEnterprise(y, e) \wedge$
$hasContact(y, c) \wedge createsObj_t(x, y, t) \wedge$
$(\exists z, t1)(CustomerRelationship(z) \wedge TimeInstant(t1) \wedge hasSupplier(z, e) \wedge hasCustomer(z, c) \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow DOLCE : presentAt(z, t2))) \rightarrow$
$addsContactRelationship_t(x, z, y, t)$

(A120) $AcquiringContact(x) \wedge ContactRelationship(y) \wedge hasContactHolderEnterprise(y, e) \wedge$
$hasContact(y, c) \wedge createsObj_t(x, y, t) \wedge \neg(\exists z, t1)(CustomerRelationship(z) \wedge TimeInstant(t1) \wedge$
$hasSupplier(z, e) \wedge hasCustomer(z, c) \wedge$
$(\forall t2)(TimeInstant(t2) \wedge t1 \leq t2 < t \rightarrow DOLCE : presentAt(z, t2))) \rightarrow$
$(\exists z)(CustomerRelationship(z) \wedge hasSupplier(z, e) \wedge hasCustomer(z, c) \wedge$
$hasContactRelationship(z, y, t) \wedge createsObj_t(x, z, t))$

O-CREAM-v2 explicitly provides some specific types of information processing activities that frequently occur in the considered domain, such as *creating invoices*, *reporting*, *performing statistical anal-*

[20] The properties $hasSupplier$, $hasCustomer$ and $hasContactRelationship$, occurring in the two following axioms, express the link between a customer relationship and its elements that play the roles of supplier, customer and contact relationship, respectively. The predicate $addsContactRelationship_t(x, z, y, t)$ infomally means that *at time t, the activity x adds to the customer relationship z a new contact relationship y.*

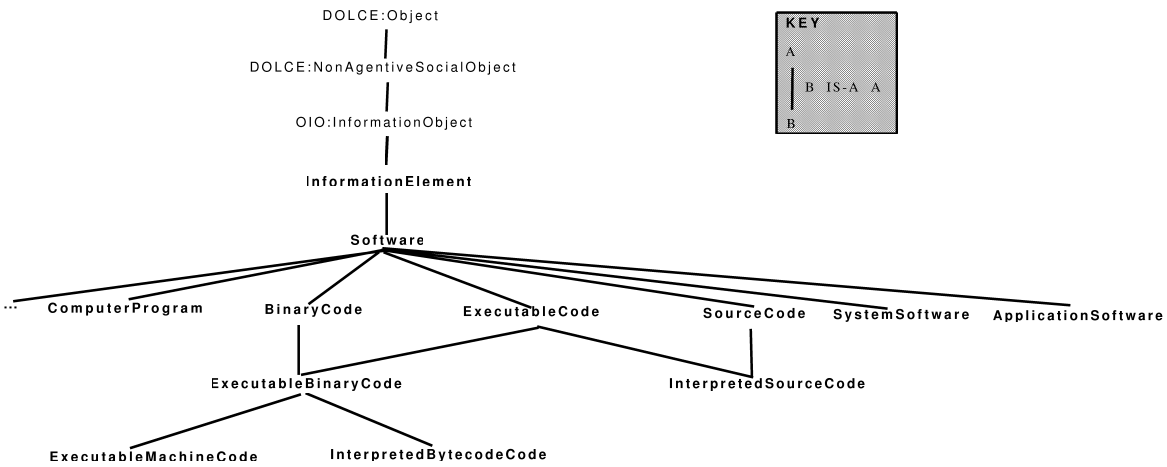

Fig. 8. A fragment of O-CREAM-v2 Software Taxonomy

yses, *modifying master files*, *performing customer segmentation*, etc. Because of the space constraints, we cannot describe them here.

A set of subclasses of $Method$ have been introduced in O-CREAM-v2, among them we have the following (whose names give an intuition of their meaning): $PaymentMethod$, $CreditCardPaymentMethod$, $BankTransferPaymentMethod$, $DeliveryMethod$, $DataAnalysisMethod$, $StatisticalDataAnalysisMethod$, $ClusteringDataAnalysisMethod$, $DataMiningMethod$ etc.

4.4. Software (Upper Core)

Software applications are the core of a set of enabling technologies for CRM. In this section we present the main concepts at relations that allow the CRM practitioners to talk about several aspects of their adopted or needed software applications, as well as the CRM software providers to describe their software solutions. Firstly, we define and characterize our notion of software and we mention some specific kinds of software categories (for which we omit here the formal characterization); then we illustrate the primitives for specifying which activities a piece of software can support or perform, the delivery models according to which it is deployed and the licences under which it is released. We conclude this section by presenting the main concepts and predicates for talking about the software resources and the hardware elements that a piece of software requires in order to work.

Given their generality, we place all of the concepts and relations relevant to software into the upper core of our ontology. Figure 8 depicts a fragment of O-CREAM-v2 Software Taxonomy, while Figure 9 reports the main properties that characterize the software elements. Since any software element is a particular kind of information element (see below), several properties that characterize the information elements in general are also exploited for software characterization.

Moreover, in this section we also mention some language types, namely: $ComputerLanguage$, $GeneralPurposeProgrammingLanguage$, $HighLevelProgrammingLanguage$, $LowLevelProgrammingLanguage$, $MachineLanguage$, $ByteCodeLanguage$. Their full formal characterization is beyond the scope of our ontology, however all of them are classified into the *Language Taxonomy*, contained in the Miscellaneous module (Section 4.5).

In Oberle *et al.* [2006] the notion of software is analyzed from an ontological point of view and an ontology for software is presented, which is based on DOLCE, DnS and OoP. The ontological analysis discussed in that paper yielded several results that have guided the development of the module modeling software-related concepts in O-CREAM-v2. Another ontology that has influenced the definition of this module is COPS (Core Ontology of Programs and Software) Lando *et al.* [2007], which provides an ontological view of computer programs and software in the frameworks provided by more abstract ontologies

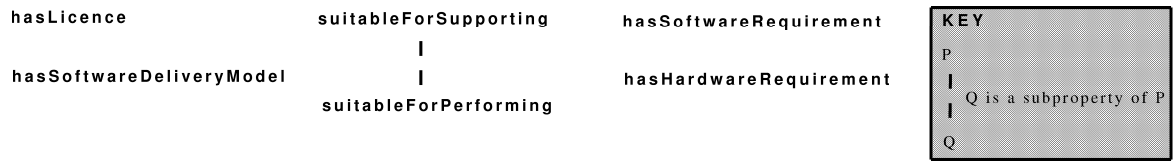

Fig. 9. Most relevant properties characterizing O-CREAM-v2 software (besides those characterizing the information elements in general)

(including both DOLCE and the I&DA core ontology for Information and Discourse Acts Fortier and Kassel [2004]).

In particular, O-CREAM-v2 has borrowed from the Core Software Ontology (CSO) Oberle *et al.* [2006] the fundamental characterization of software as a particular information object expressing a plan. Differently from CSO, and similarly to COPS, our notion of software encompasses both source and binary code programs and it admits also physical realizations other than those depending on hardware (e.g., a source code that is physically realized by only a paper listing is considered a piece of software in our framework, which is not the case in CSO). As in COPS, we introduce both the notions of software and of computer program (differenlty from COPS, we consider a computer program as a specific kind of software) and we explicitly specify the language types that can order these kinds of information elements.

In the business domain, software applications often include not only computer programs, but also other types of information elements, such as database schemas, configuration and layout data, and so on. However, a collection of such kinds of information elements without any computer program can not be considered a piece of software. Therefore, we specify that a software entity must have at least one computer program among its parts. Moreover, each piece of software can be ordered only by computer languages, which means, in particular, that the software documentation (manuals, UML models and any other document describing software) are not considered to be software. This is another difference with COPS, where the notion of *Library of Programs* is present as a software superclass. Any Library of Programs in COPS contains "Collection of Programs and, potentially, other documents (such as manuals)" Lando *et al.* [2007].

The following definition formalizes our notion of software:

(D10) $Software(x) \equiv InformationElement(x) \wedge$
$(\forall t)(DOLCE : presentAt(x,t) \rightarrow$
$(\exists p)(ComputerProgram(p) \wedge DOLCE : part(x,p,t)) \wedge$
$(\forall l)(orderedBy(x,l,t) \rightarrow ComputerLanguage(l)))$

In O-CREAM-v2, a computer program is a particular kind of software expressing at least one algorithm and ordered only by (one or more) general purpose programming language. This is expressed by the following definition:

(D11) $ComputerProgram(x) \equiv Software(x) \wedge$
$(\forall t)(DOLCE : presentAt(x,t) \rightarrow$
$(\exists a,l)(Algorithm(a) \wedge OIO : expresses(x,a,t) \wedge$
$GeneralPurposeProgrammingLanguage(l) \wedge orderedBy(x,l,t)) \wedge$
$(\forall l)(orderedBy(x,l,t) \rightarrow GeneralPurposeProgrammingLanguage(l)))$

An algorithm is a particular kind of $OoP : Plan$:

(A121) $Algorithm(x) \rightarrow OoP : Plan(x)$

It should be noted that the definitions above admit that, at different times, the same piece of software may contain different computer programs and it may be ordered by different computer languages; similarly at different times the same computer program may express different algorithms and it may be ordered by different general purpose programming languages. This formalization reflects a point of view admitting that software applications and computer programs can be maintained and modified without necessarily losing their identity.

From the two definitions above, Axiom $A40$ and the DOLCE axiom $DOLCE : part(x, y, t) \rightarrow DOLCE : presentAt(x, t) \wedge DOLCE : presentAt(y, t)$ (Axiom Ad17 in Masolo *et al.* [2003]), it follows that any software entity (when it is present) expresses at least one algorithm:

(T4) $Software(x) \wedge DOLCE : presentAt(x, t) \rightarrow$
$(\exists a)(Algorithm(a) \wedge OIO : expresses(x, a, t))$

and, given also Axiom $A32$, we have that any software entity (when it is present) is always ordered by at least one general purpose programming language (which means that there is at least one part of any software entity which is entirely ordered by such a type of language):

(T5) $Software(x) \wedge DOLCE : presentAt(x, t) \rightarrow$
$(\exists l)(GeneralPurposeProgrammingLanguage(l) \wedge orderedBy(x, l, t))$

Moreover, in 0-CREAM-v2 the temporary parthood between objects is assumed to be antisymmetric (see Section 4.5). Therefore, given that both software entities and computer programs are objects, from the two definitions above, Axiom $A165$ and the DOLCE definition of Temporary Proper Part (Definition Dd20 in Masolo *et al.* [2003]), we derive that any piece of software that is not a computer program by itself necessarily has a proper part which is a computer program:

(T6) $Software(x) \wedge \neg ComputerProgram(x) \wedge DOLCE : presentAt(x, t) \rightarrow$
$(\exists p)(ComputerProgram(p) \wedge DOLCE : properPart(x, p, t))$

Besides the two classes $Software$ and $ComputerProgram$, O-CREAM-v2 accounts for other specific kinds of software. Some basic distinctions are based on the different kinds of languages that can order a piece of software (e.g., *source code* vs *binary code*) or on the possibility of executing it either directly by hardware or by software interpretation (e.g., *executable machine code* vs *interpreted bytecode or source code*). Other basic software categories allow one to distinguish, for instance, between *system software* (such as operating systems) and *application software*(such as business software). Figure 8 reports some of these categories. Even though O-CREAM-v2 does provide a formal characterization for a few of them, we do not report it here.

In the considered domain, several further aspects of software play an important role. For the sake of brevity, we do not discuss all of them and we limit our description to some of the most interesting ones from a modelling perspective. In the following, we introduce the primitives that O-CREAM-v2 puts at disposal to specify: The kinds of activities a piece of software may perform or, at least, support; the software modules and hardware elements it requires; the models according to which it is delivered; the licences under which it is released.

In the CRM domain, as in many other domains, the execution of an activity may be supported by a software application. Thus it is important to be able to specify which activities a software application may support. To this aim, we introduce the predicate $suitableForSupporting(x, y, t)$, whose intuitive meaning is that *during time t, the piece of software x is suitable for supporting the activity y* (which does not mean that y is actually supported by x, but only that it could be):

(A122) $suitableForSupporting(x, y, t) \rightarrow$
$Software(x) \wedge Activity(y) \wedge DOLCE : TimeInterval(t)$

In some cases, a piece of software can completely automate the execution of an activity. In order to specify which activities a piece of software may perform, we introduce the predicate $suitableForPerforming(x, y, t)$, whose intuitive meaning is *during time t, the piece of software x is suitable for performing the activity y* (which does not mean that y is actually a software activity resulting from the execution of x, but only that it could be).

The suitability of a software application for performing an activity entails its suitability for supporting the same activity:

(A123) $suitableForPerforming(x, y, t) \rightarrow suitableForSupporting(x, y, t)$

If an activity x is the execution of a piece of software y, then y is suitable for performing x for the whole duration of x:

(A124) $executionOf(x, y) \wedge startsAt(x, t1) \rightarrow$
$(\forall t)((TimeInstant(t) \wedge t1 \leq t \wedge (\forall t2)(endsAt(x, t2) \rightarrow t \leq t2)) \rightarrow$
$suitableForPerforming(y, x, t))$

The following axiom links the "suitability for supporting" to the "suitability for performing" by stating that if a piece of software x is suitable for supporting an activity y, then either x is suitable for performing y or y is a complex activity and x is suitable for performing a sub-activity of y:

(A125) $suitableForSupporting(x, y, t) \rightarrow$
$suitableForPerforming(x, y, t) \vee (ComplexActivity(y) \wedge (\exists z)(DOLCE : properPart(y, z) \wedge$
$suitableForPerforming(x, y, t)))$

The converse of the axiom above holds only if the piece of software x is suitable for performing the whole activity y (as Axiom $A123$ states), but not in the general case. In fact, it seems too restrictive stating that whenever a piece of software is suitable for performing a sub-activity of a complex activity it is also suitable for supporting the complex activity as a whole. For instance, one might not want to state that a piece of software for simulating a calculating machine is suitable for supporting the activity of carrying out a statistical analysis on the sales, even though someone may use that calculating machine software application in such a statistical sale analysis.

In order to work, a piece of software may require other software resources. A software requirement may take the form of a specification of a software type (e.g., "the software application x requires OpenOffice.org Base, version 2.0 or later"), meaning that any piece of software of the specified type satisfies the requirement. Sometimes a requirement provides a very specific software type characterization, which identifies a particular individual piece of software (e.g., "the software application x requires the Java Platform Standard Edition Runtime Environment 6 for Linux x64"). In other cases a requirement actually specifies a set of alternatives, with the meaning that at least one of them should be satisfied (e.g., "the software application x requires either Microsoft IE version 7.0 or later or Firefox version 3.0 or later or Safari version 3.0 or later").

O-CREAM-v2 currently provides a very basic formal framework for representing the software requirements: the predicate $hasSoftwareRequirement(x, y, t)$ is provided, which intuitively means that *during time t, the particular x has a software requirement specified by y*. Since, in principle, besides software entities, other types of particulars can have software requirements, the argument x in the predicate above is not restricted to software. For what has been stated above, y can be either a single software requirement or a set of alternative software requirements:

(A126) $hasSoftwareRequirement(x, y, t) \rightarrow$
$DOLCE : Particular(x) \wedge$
$(SoftwareRequirement(y) \vee SoftwareRequirementAlternatives(y)) \wedge$
$DOLCE : TimeInterval(t)$

In this basic characterization, a software requirement is a $DnS : Description$ defining a software type:

(A127) $SoftwareRequirement(x) \rightarrow DnS : Description(x) \wedge$
$(\exists st)(SoftwareType(st) \wedge DnS : defines(x, st))$

A software type is a $DnS : Concept$ that classifies only software individuals (and at least one):

(A128) $SoftwareType(x) \rightarrow DnS : Concept(x)$

(A129) $SoftwareType(x) \wedge DnS : classifies(x, y, t) \rightarrow Software(y)$

(A130) $SoftwareType(x) \wedge DOLCE : presentAt(x, t) \rightarrow$
$(\exists y)(Software(y) \wedge DnS : classifies(x, y, t))$

A set of alternative software requirements is specified as a $DnS : Description$ which has at least two software requirements as proper parts:

(A131) $SoftwareRequirementAlternatives(x) \rightarrow DnS : Description(x) \wedge$
$(\exists sr1, sr2)(SoftwareRequirement(sr1) \wedge SoftwareRequirement(sr2) \wedge sr1 \neq sr2 \wedge$
$(\forall t)(DOLCE : presentAt(x,t) \rightarrow (DOLCE : properPart(x, sr1, t) \wedge DOLCE : properPart(x, sr2, t))))$

Apart from software requirements, a particular may have also hardware requirements: we model them similarly to the sofware requirements, with the only main difference that a single hardware requirement, besides specifying a hardware type (e.g., "the piece of software x requires a 64-bit, four cores processor"), can also provide a characterization for sets of hardware individuals: For instance, a requirement such as "the piece of software x requires at least 8 GB RAM" can be fulfilled by any set of RAM modules with an overall storage capacity of at least 8 GB RAM.

Software applications may be delivered according to different models. Currently, we distinguish between two different software delivery models: on-premises software and Software as a Service (SaaS). According to the first approach, the piece of software is installed and runs on a computing system of the user. In the second case, the piece of software is hosted by a provider and the user remotely access the software services (usually via Web). The predicate $hasSoftwareDeliveryModel(x, y, t)$ in O-CREAM-v2 specifies that *during time t, the piece of software x is delivered according to the software delivery model y. A software delivery model is a special kind of $DnS : Description$.*

Analogously to the delivery models, the licences under which a software application is released are modeled as special kinds of $DnS : Descriptions$ and the association between a software application and a licence is expressed by the predicate $hasLicence(x, y, t)$. We have not investigated the relationships between the software licences and the software delivery models, yet.

4.5. Miscellaneous (Upper Core)

The Miscellaneous module contains a set of concepts and relations that are not central to the modeled domain, but are used within O-CREAM-v2 and support its formal apparatus. For these items, O-CREAM-v2 only provides a light formal characterization. Unfortunately, due to the space contraints, we cannot report here this formal characterization. However, for the sake of completeness, in this section we do briefly informally introduce such concepts and relations.

First of all, this module contains a *taxonomy of languages* that extends the one introduced in Lando *et al.* [2007]. This taxonomy basically distinguishes from natural and formal languages and from visual and auditive languages. Among the formal languages, we have the computer languages, which encompass, among others, the digital formats and the programming languages. Besides these basic categories, this taxonomy contains several other kinds of languages. In the present article, we have used some of them for characterizing the notion of $Software$ and some other notions related to it (Section 4.4). However, other concepts relevant to languages emerged during the domain analysis (Section 2.1) and, consequently, have been inserted into the taxonomy. For instance, the need of dashboard-based reports and the capability of CRM software of producing them was stressed both by SME managers and ICT salesmen; this suggested to us to insert into the Language Taxonomy the notion of dashboard-based language as a specific kind of diagrammatic language, which is, in its turn, a special type of visual language.

A second taxonomy in the Miscellaneous module is the Hardware Taxonomy, which contains the main concepts related to *hardware*: CPU, storage and I/O device, hardware computing system, such as computer, smartphone, etc. These concepts have been used in the formal characterization of some O-CREAM-v2 concepts, such as $SoftwareActivity$ (Section 4.3.1) and make it possible to specify hardware requirements (Section 4.4). However, a formal characterization of these concepts (besides their classification into a taxonomy) is out of the scope of O-CREAM-v2.

Furthermore, the characterizations of several concepts in the CRM domain involve a notion of *set*. For instance: (1) an offer relationship may offer sets of particulars (Section 4.1.2); (2) a report on sales may be

about sets of sales relationships (Section 4.2.2); (3) a contact master data represents, as a whole, a set of contact relationships (Section 4.2.2); (4) a document classification activity creates a piece of information expressing the membership of a document to a set of documents (Section 4.3.1).

Unfortunately, in DOLCE, the category of sets is specified as a special kind of abstract, but no further characterization is provided for it. Moreover, at least to our knowledge, there is no extension of DOLCE providing some support for the sets, suitable for our representation needs.

It is worth mentioning that in Bottazzi *et al.* [2006] the authors introduce the notion of *Collection* as a specialization of the $DOLCE : SocialObject$ category. However collections are different from sets, as explained by the authors, and they are not suited to our needs.

For these reasons, the Miscellaneous module contains a light characterization of the notion of set. The aim is not to include in O-CREAM-v2 a whole theory of sets, thus only those aspects of sets that proved to be necessary to properly characterize other concepts of the ontology have been captured. In particular, this module includes a light characterization for: the membership relationship between a set and an element; the notion of empty set; the equivalence, inclusion and disjointness relationships between sets; the notion of characterizing property; some specific ways in which an information element can identify or represent a set (i.e., essentially, by identifying or representing each element in its extension or by intensionally representing the set via a characterizing property).

There are also other concepts that are used within O-CREAM-v2, but that were not in the focus of our ontological investigation, such as $Organization$, $Enterprise$, $HumanPerson$, $Service$ and $AmountOfMoney$ (just to list those mentioned in the present article). All of these concepts have been grouped in the Miscellaneous module, where no formal characterization is specified for them, besides their classification within the DOLCE taxomomy.

Finally, it is worth mentioning that the Miscellaneous module contains a formal restriction on objects, which specifies that the temporary parthood relationship between the objects accounted for O-CREAM-v2 is antisymmetric (while the antisymmetry of the parthood relationship does not hold in general for the DOLCE objects).

5. Discussion and Conclusions

We started this paper by considering the peculiar characteristics of the CRM domain, which suggested to us that all the actors involved in CRM activities could take significant advantages from a shared semantic model of CRM, like O-CREAM-v2. In turn, the proposed CRM semantic model would greatly benefit from the integration with an ontology of organizations, like, for instance, the one proposed in Bottazzi and Ferrario [2009], that formalizes concepts such as roles and norms underlying organizational settings.

We conclude this paper by sketching two possible exploitations of O-CREAM-v2, one from the perspective of SME that aim at implementing an efficient CRM strategy, and the other from the point of view of software houses that offer ICT solutions supporting CRM.

One of the problems that SME usually have to face when deciding to adopt some CRM strategy is how to choose the suited software support. We think that a mediation between the supply and demand of CRM-related software tools would provide a valuable help to SME that are trying to orient themselves in the CRM software market. In order to face this challenge, we designed ARNEIS (Advanced Repository for Needs of Enterprises and Innovative Software), a framework enabling intelligent Web-based repositories storing descriptions of software products and services. Moreover, we developed a prototype in order to evaluate the framework in the CRM domain Goy *et al.* [2008]. The basic idea underlying ARNEIS is that SME that decide to improve their technological integration and business automation, but lack the know-how to find the most suited ICT solution that fits their needs, can contact the ARNEIS-based repository and describe their requirements. The system matches these requirements with the descriptions of the CRM tools, uploaded in the repository by software houses, and suggests the SME the most suited solutions. The matching is possible since the descriptions of both SME requirements and CRM software tools are translated into a semantic representation, thus enabling the application of reasoning techniques

In order to check the feasibility of this approach, we built an OWL (http://www.w3.org/TR/owlfeatures/) version of a significant fragment of O-CREAM-v2, usable within the prototype based on the ARNEIS framework. A first version of the matching mechanism is described in Goy and Magro [To appear], together with the results of a preliminary evaluation of the recommendation of CRM solutions to SME, supported by the prototype.

One of the main challenges of the exploitation of a formal ontology like O-CREAM-v2 within a system based on the ARNEIS framework is providing users (both software houses and SME) with a user-friendly Web-based interface enabling them to describe their products or requirements without being exposed to the complexity of a formal semantic language such as OWL. To this purpose, we designed a form-based user interface for software houses, described in Goy and Magro [2011], and a preliminary version of a user interface based on Business Process Modeling techniques for SME, described in Goy and Magro [To appear].

The idea of supporting companies that are looking for CRM tools is obviously not new: there are many Web sites that offer evaluation and comparison services aimed at helping users to find the most suited CRM solution. However, these sites provide very simple mechanisms: for example, the CRM Software Comparison site by CRM-Reviews.com (http://www.crm-reviews.com/compare-vendors/), or the Click-by-click CRM comparison by Dovarri Inc. (http://www.dovarri.com/compare-crm.html) offer very simple comparison tables including a limited number of CRM solutions, compared on the basis of a small pre-defined set of features. Other sites offer recommender services, but, again, they are based on a extremely limited set of pre-defined characteristics (e.g., the CRM Solution Advisor by Compare CRM: http://www.comparecrm.com/crm/crm-vendor-recommendations.php; the Vendor Guru by Elite CRM Software: http://www.elitecrmsoftware.com/crm.html). With respect to sites like these ones, our approach aims at offering SME a much more detailed way to describe their needs; moreover, we enable also CRM vendors to describe their software solutions, through a Web-based user interface (see Goy and Magro [2011]). Thus, in our approach, both the CRM solutions available and the features on which the comparison is based are dynamic and not pre-defined, as in the mentioned examples. Moreover, the knowledge about CRM on which our prototype is based is far more complex than a simple feature list; this complexity should ensure a deeper understanding of both the vendor offer and the SME needs, and thus a more grounded recommendation.

Web-based repositories exploiting the ARNEIS framework can be very useful for SME, to help them in finding the most suited software support for their CRM activities, but also for software houses, that can exploit the repository to promote their products. Within this perspective, the availability of a formal semantic representation of CRM-related concepts like O-CREAM-v2 could be useful for software houses in order to support them in building formal semantic descriptions of their product and services. Such semantic descriptions could be indexed in semantic search engines Mangold [2007] and ontology-based Information Retrieval (IR) systems Tran *et al.* [2007], thus offering software houses an important marketing opportunity.

References

R. Arndt, R. Troncy, S. Staab, and L. Hardman. Comm: A core ontology for multimedia annotation. In S. Staab and R. Studer, editors, *Handbook on Ontologies, Second Edition*, pages 403–421. Springer, 2009.

S. Borgo and P. Leitão. The role of foundational ontologies in manufacturing domain applications. *LNCS*, 3290:670–688, 2004.

S. Borgo and C. Masolo. Foundational choices in dolce. In S. Staab and R. Studer, editors, *Handbook on Ontologies, Second Edition*, pages 361–381. Springer, 2009.

S. Borgo and L. Vieu. From physical artefacts to products. In *Proc. Second Workshop FOMI*, pages 85–99, 2006.

E. Bottazzi and R. Ferrario. Preliminaries to a dolce ontology of organizations. *International Journal of Business Process Integration and Management*, 4(4):225–238, 2009.

E. Bottazzi, C. Catenacci, A. Gangemi, and J. Lehmann. From collective intentionality to intentional collectives: an ontological perspective. *Cognitive Systems Research - Special Issue on Cognition Joint Action and Collective Intentionality*, 7(2-3):192–208, 2006.

A. Devalle. Customer relationship management. In *V. Cantino et al. Management Information Systems*, pages 343–368. McGraw-Hill, 2005.

J. Dychi. *The CRM Handbook: a business guide to customer relationship management*. Addison-Wesley, 2001.

J.-Y. Fortier and G. Kassel. Managing knowledge at the information level: an ontological approach. In *Proc. ECAI 2004 Workshop on Knowledge Management and Organizational Memories*, pages 39–45, 2004.

J. Freeland. *The Ultimate CRM Handbook*. McGraw-Hill, 2005.

A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. *LNCS*, 2888:689–706, 2003.

A. Gangemi, S. Borgo, C. Catenacci, and J. Lehmann. *Task Taxonomies for Knowledge Content*. 2005. Metokis Deliverable D07.

A. Goy and D. Magro. Managing user interaction in an ontology-based system. In *Proc. 7th International Conference on Web Information Systems and Technologies*. INSTICC Press, 2011.

A. Goy and D. Magro. How semantic web technologies can support the mediation between supply and demand in the ICT market: the case of customer relationships management. In I. Bedini, F. D. Dorloff, and E. Kajan, editors, *Handbook of Research on E-Business Standards and Protocols: Documents, Data and Advanced Web Technologies*. IGI Global, To appear.

A. Goy, D. Magro, and F. Prato. ARNEIS: A web-based intelligent repository of ict solutions for e-business. In *Proc. iiWAS2008*, pages 403–406, 2008.

M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. *LNCS*, 5268:329–346, 2008.

P. Lando, A. Lapujade, G. Kassel, and F. Fürst. Towards general ontology of computer programs. In *Proc. ICSOFT (PL/DPS/KE/MUSE)*, pages 163–170, 2007.

D. Magro and A. Goy. The business knowledge for customer relationship management: an ontological perspective. In *Proc. OBI2008*. ACM Press, 2008.

D. Magro and A. Goy. Towards a first ontology for customer relationship management. In *Proc. CSTST'2008*, pages 637–643. ACM, 2008.

C. Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34, 2007.

C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. *WonderWeb deliverable D18*. ISTC-CNR, 2003. Technical Report.

D. Oberle, S. Lamparter, S. Grimm, D. Vrandečić, S. Staab, and A. Gangemi. Towards ontologies for formalizing modularization and communication in large software systems. *Applied Ontology*, 1(2):163–202, 2006.

P. Rittgen, editor. *Handbook of Ontologies for Business Interaction*. Idea Group, 2007.

T. Tran, S. Bloehdorn, P. Cimiano, and P. Haase. Expressive resource descriptions for ontology-based information retrieval. In *ICTIR 2007 (First International Conference on the Theory of Information Retrieval)*, pages 55–68. Springer, 2007.

M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13(1):31–89, 1998.

W3C. OWL Web Ontology Language current status. http://www.w3.org/standards/techs/owl#w3c_all, 2011.