International Journal of Digital Information and Wireless Communications (IJDIWC) 1(1): 146-160
The Society of Digital Information and Wireless Communications, 2011(ISSN 2225-658X)

A conceptualization of organizations involved in Product Design: a first step towards reasoning and knowledge management

Yishuai Lin, Vincent Hilaire, Nicolas Gaud, and Abderrafiaa Koukam

Laboratoire Syst`emes et Transports (SeT)
Universit´e de Technologie de Belfort-Montb´eliard (UTBM)
90010 Belfort cedex, France
vincent.hilaire@utbm.fr
+33 384 583 009
www.multiagent.fr

ABSTRACT

The aim of this paper is to build an ontology of organizations.
This ontology will be used to analyze, reason and understandorganizations. The targeted organizations are those composed of individualsinvolved in the design of a product and, to do so, following a design process. This ontology will be used to support knowledge managementwithin the described organizations. More specifically, the ontology willprovide means for reasoning, annotating resources, monitoring design processes, enabling searches and proactively proposing tips and proper content. The presented approach is based upon the use of an existing organizational metamodel, namely CRIO, already used for the description of Multi-Agent Systems (MAS) organizations. In our case, the concepts of this metamodel are used to model human activities.

Keywords: Ontology, Organization modeling, Knowledge Management

1 Introduction The aim of this paper is to build an ontology of organizations. This ontology will be used to analyze, reason and understand about organizations. Indeed, as stated in [3], an ontology of organizations ”is the first, fundamental and ineliminable pillar on which to build a precise and rigorous enterprise modeling”. Obviously, the field of ontologies for organizations is very broad, among the works in this domain one can cite [3, 7]. In order to reduce our field of investigation, we have followed the knowledge engineering methodology proposed in [17]. The first step of this methodology is ”Determine the domain and scope of the ontology”.

During this step, the knowledge engineer has to answer the following questions:

– What is the domain that the ontology will cover ?
– For what we are going to use the ontology ?
– For what types of questions the information in the ontology should provideanswers ?

In our case the targeted organizations are those composed of individuals involved in the design of a product. These individuals follow a previously defined design process to frame their design activities. The ontology is used to support knowledge management within the described organizations. More specifically, the ontology will provide means for reasoning, annotating resources, monitoring design processes, enabling searches and proactively propose tips and proper contents.

The presented approach is based upon the use of an existing organizational metamodel, namely CRIO [5], already used for the description of Multi-Agent Systems (MAS) organizations [6]. In our case, the concepts of this metamodel are used to model human activities. The concepts and relationships of this metamodel are represented by the K-CRIO ontology.

This paper is organized as follows: Section 2 reviews and discusses related works. Section 3 introduces the background of our work. Section 4 presents an ontology for CRIO Metamodel. Section 5 details some instances of this ontology taking as an example a software development organization. Eventually, Section 6 concludes.

2 Related works

Formal structures such as OWL and RDFS have been used for Personal Information Management before; the PIMO model [18] aims to represent parts of the Mental Model necessary for tasks involving knowledge. The Mental Model is part of the cognitive system of a person. Subjective to a person, the mental model is individual and cannot be externalized thoroughly. The definition for a Personal Information Model (PIMO) is given as follows: A PIMO is a Personal Information Model of one person. It is a formal representation of parts of the users Mental Model. Each concept in the Mental Model can be represented using a Thing or a subclass of this class in RDF. Native Resources found in the Personal Knowledge Workspace can be categorized, and are occurrences of Thing. The vision is that a Personal Information Model reflects and captures a user's personal knowledge, e.g., about people and their roles, about organizations, processes, and so forth, by providing the vocabulary (concepts and their relationships) for expressing concepts as well as concrete instances. In other words, the domain of a PIMO is meant to be ”all things and native resources that are pertinent to the user when doing work involving knowledge”. Though ”native” information models and structures are widely used, there is still much potential for a more effective and efficient exploitation of the underlying knowledge. Compared to the cognitive representations humans build, there are mainly two shortcomings in native structures:

– Model richness: current state of cognitive psychology assumes that humans build very rich models, representing not only detailed factual aspects, but also episodic and situational information. Native structures are mostly taxonomyoriented or keyword-oriented.

– Models coherence: though nowadays (business) life is very fragmented, humans tend to interpret situations as a coherent whole and have representations of concepts that are comprehensive across contexts. Native structures, on the other hand, often reflect the fragmentation of multiple contexts. They tend to be redundant (i.e., the same concepts at multiple places in multiple native structures). Frequently, inconsistencies are the consequence.

In brief, the PIMO shall mitigate the shortcomings of native structures by providing a comprehensive model on a sound formal basis.

Multilayered Semantic Social Network (MSSN) Model [4] proposes a multilayered semantic social network model that offers different views of common interests underlying a community of people, which is working within an ontologybased personalization framework [20], user preferences are represented as vectors ui = (ui,1, ui,2, ..., ui,N) where the weight ui,j 2 [0, 1] measures the intensity of the interest of user i for concept cj in the domain ontology, N being the total number of concepts in the ontology. Similarly, the objects dk in the retrieval space are assumed to be described (annotated) by vectors (dk,1, dk,2, ..., dk,N) of concept weights, in the same vector-space as user preferences. Based on this common logical representation, measures of user interest for content items can be computed by comparing preference and annotation vectors, and these measures can be used to prioritize, filter and rank contents (a collection, a catalog, a search result) in a personal way. The applicability of the proposed model to a collaborative filtering system is empirically studied. Starting from a number of ontology-based user profiles and taking into account their common preferences, the concept space domain is automatically clustered. With the obtained semantic clusters, similarities among individuals are identified at multiple semantic preference layers, and emergent, layered social networks are defined, suitable to be used in collaborative environments and content recommenders.

The AIC Model represents such a system as a tripartite graph with hyperedges [14]. The set of vertices is partitioned into the three (possibly empty) disjoint sets A = {a1, a2, ..., ak}, C = {c1, c2, ..., ci}, I = {i1, i2, ..., im} corresponding to the set of actors (users), the set of concepts (tags, keywords) and the set of annotated objects (bookmarks, photos etc). It extends the traditional bipartite model of ontology (concepts and instances) by incorporating actors in the model. In a social tagging system, users tag objects with concepts, creating ternary associations between the user, the concept and the object. Thus the folksonomy is defined by a set of annotations T 2 A × C × I. Such a network is most naturally represented as an hypergraph with ternary edges, where each edge represents the fact that a given actor is associated with a certain instance and a certain concept. In particular, the author defines the representing hypergraph of a folksonomy T as a (simple) tripartite hypergraph H(T) = (V,E) where V = A [C [I, E = {{a, c, i} | (a, c, i) 2 T}.

The MOISE+ Model [10] structure is built up in three levels: one is the behaviors that an agent playing a role is responsible for (individual), the other is the structure and interconnection of the roles with each other (social), and the last is the aggregation of roles in large structures (collective). In MOISE+, as in MOISE, three main concepts, roles, role relations, and groups, are be used to build, respectively, the individual, social, and collective structural levels of an organization. Furthermore, the MOISE original structural dimension is enriched with concepts such as inheritance, compatibility, cardinality, and sub-groups.

– Individual level is formed by the roles of the organization. A role means a set of constraints that an agent ought to follow when it accepts to enter a group playing that role. Following, these constraints are defined in two ways: in relation to other roles (in the collective structural level) and in a deontic relation to global plans (in the functional dimension). In order to simplify the specification, like in Object-Oriented (OO) terms, there is an inheritance relation among roles. If a role p0 inherits a role p (denoted by p p0), with p 6= p0, p0 receives some properties from p, and p0 is a sub-role, or specialization, of p. In the definition of the role properties presented in the sequence, it will be precisely stated what one specialized role inherits from another role. For example, in the soccer domain, the attacker role has many properties of the player role (pplayer pattacker). It is also possible to state that a role specialize more than one role, i.e., a role can receive properties from more than one role. The set of all roles are denoted by Rss.

Following this OO inspiration, we can define an abstract role as a role that cannot be played by any agent. It has just a specification purpose. The set of all abstract roles is detonated by Rabc(Rabc Rss). There is also a special abstract role Psocwhere 8(p2 Rss)Psoc P, trough the transitivity of , all other roles are specializations of it.

– Social level is when the inheritance relation does not have a direct effect on the agents’ behavior, there are other kinds of relations among roles that directly constrain the agents. Those relations are called links and are represented by the predicate link(ps,pd,t) where ps is the link source, pd is the link destination, and t 2 acq, com, aut is the link type. In case the link type is acq (acquaintance), the agents playing the source role ps are allowed to have a representation of the agents playing the destination role pd (pd agents, in short).In a communication link(t = com), the ps agents are allowed to communicate with pd agents. In an authority link(t = aut), the ps agents are

allowed to have authority on pd agents, i.e., to control them. An authority
link implies the existence of a communication link that implies the existence
of an acquaintance link:
linkps,pd,aut =) linkps,pd,com (1)
linkps,pd,com =) linkps,pd,acq (2)
Regarding the inheritance relation, the links follow the rules:
(link(ps,pd,t) ^ ps ps0) =) link(ps0 ,pd,t) (3)
(link(ps,pd,t) ^ ps pd0) =) link(ps,pd0 ,t) (4)
For example, if the coach role has authority on the player role link(pcoah,pplayer,aut)
and player has a sub-role (pplayer pattacker), by Eq. (4), a coach has also authority on attackers. Moreover, a coach is allowed to communicate with
players (by Eq. (1)) and it is allowed to represent the players (by Eq. (2)).
– Collective level: the links constrain the agents after they have accepted to
play a role. However, we should constrain the roles that an agent is allowed to
play depending on the roles this agent is currently playing. This compatibility
constraint pa ./ pb states that the agents playing the role are also allowed
to play the role pb (it is a reflexive and transitive relation). As an example,
the team leader role is compatible with the back player role pleader ./ pback.
If it is not specified that two roles are compatible, by default they are not.
Regarding the inheritance, this relation follows the rule
(pa ./ pb ^ pa 6= pb ^ pa v p0) =) (p0 ./ pb) (5)
Hence, there should be a series of rules and relationships defined within the
collective level.
3 Background
3.1 CRIO metamodel
The CRIO metamodel relies upon four fundamental concepts: Capacity, Role,
Interaction and Organization (see Figure 2). An organization is composed of
Roles, which are abstract behaviors interacting following defined interactions
within scenarios while executing their Role plans. An organization has a context
that is described in terms of an ontology. Roles participate to the achievement
of their organization goals by means of their Capacities.
An organization is defined by a collection of roles that take part in systematic
institutionalized patterns of interactions with other roles in a common context.
This context consists in a shared knowledge, social rules/norms, social feelings,
and it is defined according to an ontology. The aim of an organization is to
fulfill some requirements. An organization can be seen as a tool to decompose a
system and it is structured as an aggregate of several disjoint partitions. Each
organization aggregates several roles and it may itself be decomposed into suborganizations.
A Role defines an expected behavior as a set of role tasks ordered by a
plan, and a set of rights and obligations in the organization context. The goal
of each Role is to contribute to the fulfillment of (a part of) the requirements
of the organization within which it is defined. Roles use their capacities for
participating to organizational goals fulfillment; a Capacity is a specification
of a transformation of a part of the designed system or its environment. This
transformation guarantees resulting properties if the system satisfies a set of
constraints before the transformation. It may be considered as a specification of
the pre- and post-conditions of a goal achievement. This concept is a high level
abstraction that proved to be very useful for modeling a portion of the system
capabilities without making any assumption about their implementations as it
should be at the initial analysis stage.
A Capacity describes what a behavior can do or what a behavior may require
to be defined. As a consequence, there are two main ways of using this concept:
– it can specify the result of some role interactions, and consequently, the
results that an organization as a whole may achieve with its behavior. In
this sense, it is possible to say that an organization may exhibit a capacity.

– capacities may also be used to decompose complex role behaviors by abstracting and externalizing a part of their tasks into capacities (for instance by delegating these tasks to other roles). In this case, the capacity may be considered as a behavioral building block that increases modularity and reusability.

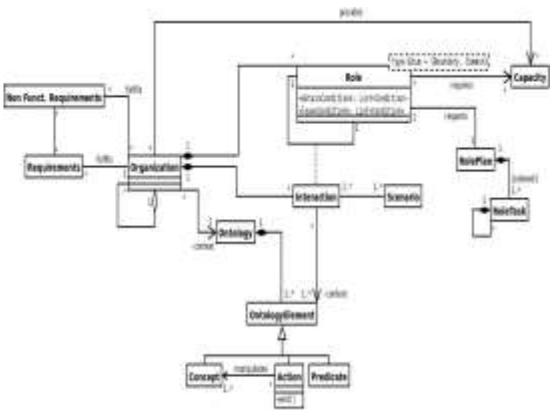

Fig. 1. Problem Domain of the CRIO metamodel

Let's take an example to illustrate these concepts. If we have a Motion control organization composed of two roles: Route requester role that asks for a route between two locations and Route provider role that provides routes. The capacity to find the shortest path in a weighted directed acyclic graph G(N, E), from a source node to a destination node d may be required by the role Route provider. This capacity may be realized in various ways, by using classical graph algorithms, if the know-how of a single entity is considered, or it can also be modeled by an organization. The Ant Colony is a well-known organization able to find a solution to the problem of finding the shortest path in a graph [1].

The solution (the shortest path) emerges from interactions among Ants in their environment. Let us suppose that the environment represents the graph G, the source node s is mapped to the Ant Hill and the destination d to a food source.

At the level n + 1, the Route Choice organization is responsible for providing the best route between two given points to another organization (for instance the Motion Control organization). This capacity provides the solution of a problem that is effectively solved at a lower level of abstraction (level n). The ant colony organization that is located at a lower level in the organizational hierarchy can realize this capacity. The capacity concept thus allows to define how an organization at level n may contribute to the behavior of a role at level n + 1.

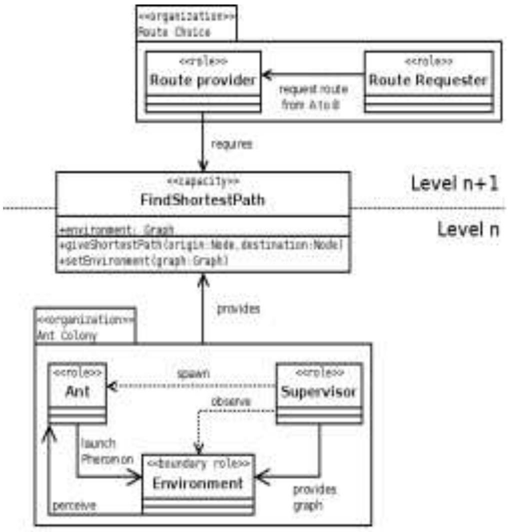

Fig. 2. Example of Organizations, Roles and Capacity diagram

3.2 Ontologies and OWL

The word ontology comes from the Greek ”ontos” for being and ”logos” for word. Generally, the term ontology is defined as ”an explicit specification of a conceptualization” [9] or ”a set of types, properties, and relationship types” [8]. Ontology defines concepts in a specific area and their relationships. Furthermore, it can be used as a set of well-defined constructs that can be used to build structured knowledge. Ontologies include rich relationships among terms, rich taxonomies, and enables intelligent researches. For humans, ontologies enable better access to information and promote knowledge reuse and shared understanding. For computers, ontologies facilitate comprehension of information and more extensive processing (Ontology Engineering). Being a technology used for KM, ontology defines the terms used to describe and represent an area of knowledge. It is established as a powerful tool to enable knowledge sharing, and a growing number of applications have benefited from the use of ontology as a means to achieve semantic interoperability among heterogeneous, distributed systems.

As a language for describing ontology, OWL stands for Ontology Web Language1.

In fact, it is a family of knowledge representation languages for authoring ontologies. OWL is supported by the World Wide Web Consortium. OWL is based upon RDF and RDFS which are extensions of XML for describing web resources.

OWL has three increasingly-expressive sub-languages: OWL Lite, OWL DL, and OWL Full. Compared with RDF, OWL has more features that allow a greater machine interpretability and comes with a larger vocabulary and richer syntax.

In the following, we introduce the logical constructs of OWL. Classes provide an abstraction mechanism for grouping resources with similar characteristics. Every OWL class is associated with a set of individuals (called the class extension). The individuals in the class extension are called the instances of the class. OWL classes are described through ”class descriptions” which can be combined into ”class axioms”. Specifically, OWL distinguishes six types of class descriptions: a class identifier (a URI reference), an exhaustive enumeration of individuals (that together form the instances of a class), a property restriction (including value constraints and cardinality constraints), the intersection of two or more class descriptions, the union of two or more class descriptions and the complement of a class description. Into class axioms, OWL contains three language constructs for combining class descriptions, which are rdfs:subClassOf, owl:equivalentClass and owl:disjointWith. In addition, OWL distinguishes between two main categories of properties that an ontology designer may want to define, which are Object properties defining relationships between classes and Datatype (properties linking individuals to data values).

Equally, OWL also supplies various types of property axioms to define additional characteristics of properties, which are as following: RDF Schema constructs (rdfs:subPropertyOf, rdfs:domain and rdfs:range), relations to other properties (owl:equivalentProperty and owl:inverseOf), global cardinality constraints

(owl:FunctionalProperty and owl:InverseFunctionalProperty), logical property
characteristics (owl:SymmetricProperty and owl:TransitiveProperty). Finally, individuals
are defined with individual axioms (also called "facts"), where are two
types of facts: one about class membership and property values of individuals
and the other about individual identity (owl:sameAs, owl:differentFrom and
owl:AllDifferent).

1 http://www.w3.org/TR/owl-features/

4 Ontology for human design processes

In the following we will try to single out which are the main entities of an ontology
to represent organizations. The K-CRIO ontology is presented in Figure 7.

The targeted organizations are those dedicated to product design. We have
then defined a concept named DesignObject (owl:class) which is the root of all
possible products that an organization can produce.

An organization can be seen as a set of interacting entities: sub-organizations
or roles, which are regulated by social rules and norms.

– With respect to the ontology, an organization may be seen as a concept
connected to other concepts by various kinds relationships, such as hierarchical
relations between organizations and sub-organizations, or composed
of relation between an organization and its roles.

– With respect to the human process in enterprises, an organization may be
considered as a collective global system able to achieve particular goals
through its collaborative members.

Using OWL, the concept of organization may be specified as following: Organization
is an owl:class which may be linked to sub-organizations with "isSubOrganizationOf"
(owl:ObjectProperty) and "includes" (owl:ObjectProperty) a collection
of Roles (an owl:class), with "provided" (owl:ObjectProperty) Capacity
(as well as an owl:class) and "hascontext" (owl:ObjectProperty) Ontology (an owl:class). Additionally, Organization "isThePlaceOf" Interactions happening.

It may be expressed as detailed in Figure 3.

```
<owl:Class rdf:ID="Organization"/>
<owl:Class rdf:ID="Ontology"/>
<owl:Class rdf:ID="Capacity"/>
<owl:Class rdf:ID="Role"/>
<owl:Class rdf:ID="Interaction"/>

<owl:ObjectProperty rdf:ID="hasContext">
  <rdfs:domain rdf:resource="#Organization"/>
  <rdfs:range rdf:resource="#Ontology"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="includes">
  <rdfs:domain rdf:resource="#Organization"/>
  <rdfs:range rdf:resource="#Role"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="provided">
  <rdfs:range rdf:resource="#Capacity"/>
  <rdfs:domain rdf:resource="#Organization"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isThePlaceOf">
  <rdfs:domain rdf:resource="#Organization"/>
  <rdfs:range rdf:resource="#Interaction"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSubOrganizationOf">
  <rdfs:domain rdf:resource="#Organization"/>
  <rdfs:range rdf:resource="#Organization"/>
</owl:ObjectProperty>
```

Fig. 3. Organization in K-CRIO

Meanwhile, we defined some necessary restrictions between Organization and
relative class as following. One Organization may have its unspecified number sub-Organizations (one or more) or not have its sub-Organization. For example, if we consider a university as one Organization, different departments may be seen as its sub-organizations. More precisely, one department is usually composed of diverse majors, which may be as sub-organizations of this department. In order to describe this restriction, we use one kind of qualified cardinality restrictions in OWL, cardinality constraints, which includes owl:cardinality, owl:minCardinality, owl:maxCardinality [19]. Hence, Organization " isSubOrganizationOf " its sub-Organizations with minCardinality 0 (owl:restriction). Moreover, one Organization must include one Role at least and in which one or more

interaction must be occurred. Describing with OWL, Organization "includes" Role with minCardinality 1 (owl:restriction) and "isThePlaceOf" interaction happening with minCardinality 1 (owl:restriction). It may be detailed in the Figure 4.

```
<owl:Class rdf:ID="Organization">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >0</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isSubOrganizationOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="includes"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isThePlaceOf"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Fig. 4. Restrictions of Organization in K-CRIO

A role may identify a person, an activity or a service which is a necessary part to achieve social objectives (goals of its organization). In order to fulfill this common target of one organization, each role as a specific individual capacity provided by the organization enabling interactions with other roles belonging to the same organization. From the view of K-CRIO, Organization (an owl:class) "includes" (owl:ObjectProperty) Roles (an owl:class) which may "required" (owl:ObjectProperty) Capacity (an owl:class). Interaction occurring in Organization "hasParticipants" (owl:ObjectProperty) that are Role.

A capacity is a know-how and ability, which may be considered as an interfacebetween the role and role-players. Capacities are required by the role and provided by the organization to define the behaviors of other roles. With the respect of K-CRIO, besides the relationship with Organization and Role described above, Capacity is represented by an owl:class. This class is related to some ContextOntologyElement (which are parts of the Organization Ontology defining it context) divided in two sets "input" and "output" (both are owl:ObjectProperty).

The Capacity class is also related to properties which are conceptualized by the Predicate concept. A predicate is an assertion on a property of some ContextOntologyElement.

There are two types of relationships between Capacity and Predicate. The first type is named "requires" and represents the properties that are required by the capacity. The second type is named "ensures" and represents the effects of the capacity.

Interactions are mandatory for modeling human processes. The aim of an interaction in this context is to achieve a goal or to contribute to a goal of one organization. Interactions may be divided into Casual Interaction and Formalized Interaction. The former ones depict interactions between roles which are not defined initially in the model and that can take place in a non determined fashion.

Inversely, formalized interactions identify how different roles belonging to the same organization can interact with each other to achieve the common goal of the organization and in the meanwhile produce DesignObjects of any sort. It may be expressed as detailed in Figure 5. Interactions may then be seen as a description of possible patterns of actions and exchanges between the participant roles that is very similar to a workflow. In order to conceptualize Formalized- Interaction, we have chosen to reuse an existing ontology, OWL-WS, dedicated to the description of workflows [2] and inspired from OWL-S [12]. OWL-WS is based on the assumption that a workflow is a kind of complex service and therefore it can be represented in OWL-WS as a full OWL-S Service. This service can be a simple one or composed of simpler services using the OWL-S control constructs such as: Sequence, RepeatUntil, Split, etc.

```
<owl:Class rdf:ID="Interaction"/>
<owl:Class rdf:ID="FormalizedInteraction">
  <rdfs:subClassOf rdf:resource="#Interaction"/>
</owl:Class>
<owl:Class rdf:ID="CasualInteraction">
  <rdfs:subClassOf rdf:resource="#Interaction"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="produces">
  <rdfs:range rdf:resource="#DesignObject"/>
  <rdfs:domain rdf:resource="#FormalizedInteraction"/>
</owl:ObjectProperty>
```

Fig. 5. Interaction in K-CRIO

A process may be an atomic process which is a description of one (possibly complex) message and returning one (possibly complex) message in response. A process may also be a composite which means that it can be divided into atomic processes describing a behavior (or a set of behaviors) sending and receiving a series of messages. If we consider the formalized interaction as a composite process owning an overall effect, the entire process must be performed in order to achieve that objective. Moreover, the definition of formalized interaction expres the different ways of executing this interaction and correlative results. Let's take the example of a selling service, represented by one organization named selling service including two roles: Client and Bank Service. The formalized interaction details how to perform this trade between these two roles (exchanging messages).

– The initial state is when the Client inputs the Credit Card Number and code, which is a message sent to Bank Service.

– When Bank Service receives a message from the Client role, there should be a control accompanying different situations as

• Both datum are right, Bank Service returns a confirm message to Client;

• One datum is invalid, Bank Service returns an error message to Client, simultaneously, the state of Client returns to original state (waiting for an input of the Credit Card Number and Password and waiting new answer of Bank Service);

– This sequence is repeated until Client receives the confirm message from Bank Service. He can then send Verify Paying message to Bank Service;

– Bank Service valid the charge following the message Verify Paying from Client.

From the point of view of K-CRIO, CasualInteraction and FormalizedInteraction are represented by two owl:class both subclasses of Interaction (an owl:class), which are related by "hasParticipants" (owl:ObjectProperty) to Roles. The FormalizedInteraction class is related by the "produces" (owl:ObjectProperty) relationships to the "DesignObject" (an owl:class) concept.

In summary, the whole K-CRIO Ontology is presented in Figures 6 and 7.

5 Example

In this section, an example of organization described with K-CRIO is presented.

In order to illustrate the K-CRIO ontology, we have chosen one project team of a company. We suppose the team is a common Software Developing Team in an IT (Information Technology) Company developing softwares. Moreover, its developing process conforms to the Waterfall Model of software development process [13] (Actually, other models of software development process may also be modeled by CRIO, like Spiral Model, Rapid Prototype Model, etc). The Waterfall Model is presented by Figure 8.

Based on the information got from [13] and Figure 8, we could see that there are five phases during the software developing process following Waterfall Model as:

Fig. 6. K-CRIO taxonomy

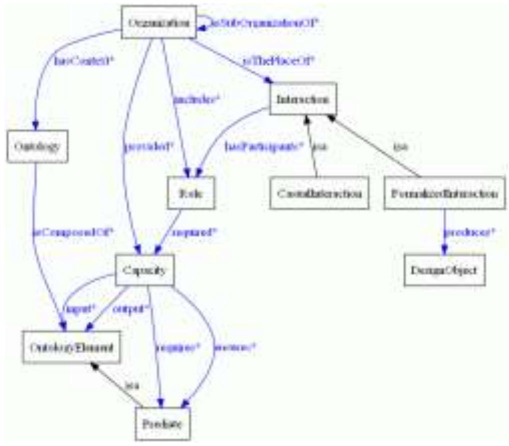

Fig. 7. K-CRIO Ontology

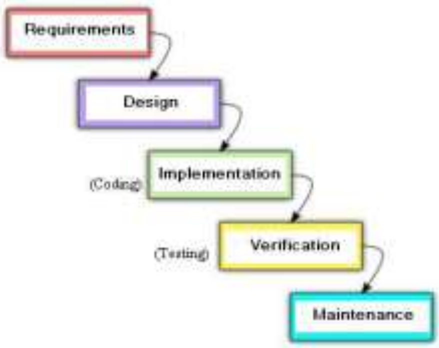

Fig. 8. Software development process: Waterfall Model

– In the Requirement phase, Requirement Analyst needs to visit the customer and studies their system requirement. They examine the need for possible software automation in the given software system. After feasibility study, the development team provides a document that holds the different specific recommendations for the candidate system. It also consists of personnel assignments, costs of the system, project schedule and target dates.

– In the Design phase, the overall software structure and its outlay are defined,
like package architecture, the database design, the data structure design, etc.
All of these could be described in the documents by designer.
– In the Implementation phase, the whole system design must be decoded into a machine-readable form by coders with programming languages like C,
C++, Pascal and Java and so on .
– After code generation phase the software program testing begins to detect
the bugs in the Verification phase.
– In the Maintenance phase, software will definitely go through change once
when it is delivered to the customer.
As the above state, we can understand how the different people work together
to achieve a software project refereed by Waterfall Model. In the following, we
will introduce specificities of the example modeled with K-CRIO. An IT Company
should be seen as an Organization and the Software Developing Team as
well as one Organization is one of its sub-organizations which includes Roles
such as: Project Leader, Requirement Analyst, Designer, Coder, Tester (System
Tester, and Unit Tester). Referencing about the actual process of a Software
Development Project, the Capacities required by the Roles are:
– managing project, organizing meeting, confirming domain, supervising schedule, making decisions, examining and checking, etc. (required by Project Leader);
– communicating with clients or customers, analyzing the requirement of users,
writing requirement document (required by Requirement Analyzer);
– designing system (including system design, architecture design and database
design, etc.), writing design document (required by Designer);
– coding (required by coder);
– test, writing test report (required by Tester, System Tester, Unit Tester);
Following K-CRIO definition of previous section, above description may be expressed
as presented in Figure 9. Additionally, we take an instance of Project
Leader to explain how to define these in OWL, which is in the Figure 10.

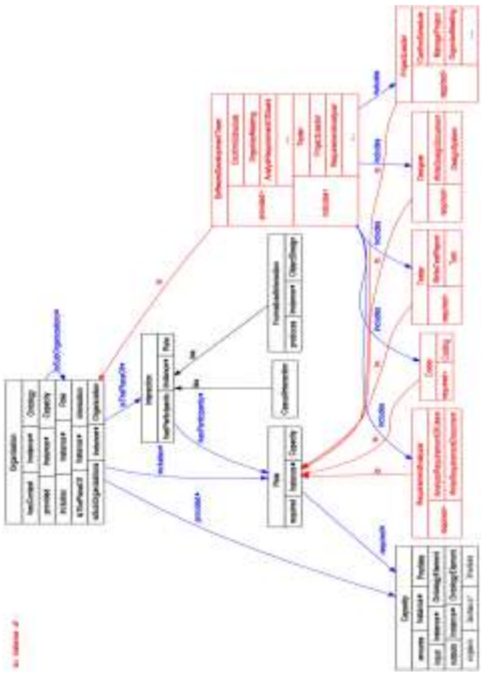

Fig. 9. An Example of K-CRIO

```
<Organization rdf:ID="SoftwareDevelopmentTeam">
   <includes rdf:resource="#ProjectLeader"/>
   <provided rdf:resource="#SuperviseSchedule"/>
   <provided rdf:resource="#ConFirmSchedule"/>
   <provided rdf:resource="#OrganizeMeeting"/>
   <provided rdf:resource="#CoFirmDomain"/>
   <provided rdf:resource="#MakeDecisions"/>
   <provided rdf:resource="#ExamineAndCheck"/>
 </Organization>

 <Role rdf:ID="ProjectLeader">
   <required>
     <Capacity rdf:ID="ConFirmSchedule"/>
   </required>
   <required>
     <Capacity rdf:ID="OrganizeMeeting"/>
   </required>
   <required>
     <Capacity rdf:ID="SuperviseSchedule"/>
   </required>
   <required>
     <Capacity rdf:ID="CoFirmDomain"/>
   </required>
   <required>
     <Capacity rdf:ID="ExamineAndCheck"/>
   </required>
   <required>
     <Capacity rdf:ID="MakeDecisions"/>
   </required>
 </Role>
```

Fig. 10. Defining Project Leader in OWL

The interactions in the SDT Organization (Software Developing Team) may be considered as Casual Interaction and Formalized Interaction separately. Chat is a kind of Casual Interaction between different persons (Roles), other examples may be Exchanging Mail or Joining Conference Meeting, etc. Following the process description fashion of OWL-WS, the whole Software Developing process may be seen as a Composite Process, which is composed of five Composite Processes by the Control Construct Sequence as a Composite Process: Requirement Analyzing, a Composite Process: System Designing, a Composite Process: Implementation, a Composite Process: Verification and a Composite Process: Maintenance. Precisely, we select one of these five composite processes, System Designing, in order to explain how the process does work.

As sketched by Figure 11, the Composite Process System Designing is a Repeat-While process with two components, a Composite Process: Review Design Document and an Atomic Process: Handover Design Document. The Condition Redo controls the loop and is initialized to true. While Redo is true, the process Review Design Document is executed and return a new value of Redo. If Condition Redo is false, the Atomic Process Handover Design Document is executed. The Review Design Document is composed of three sequential Atomic Processes: Promote Design Document, Submit Design Document and Check Design Document. With the fashion of OWL-WS, it may be expressed in the Figure 11.

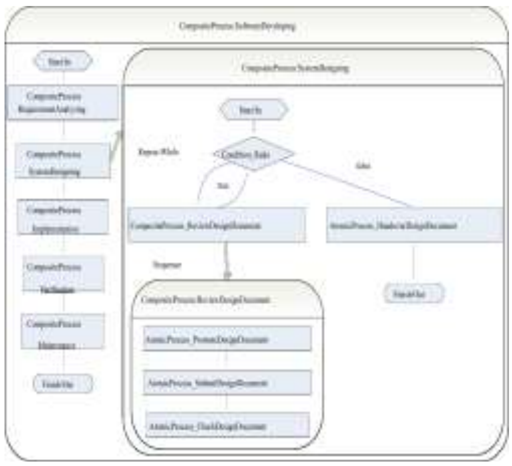

Fig. 11. System Designing: A part of Software Developing Process

Fig. 12. Expressing process in OWL-WS

Because the initial value of Condition Redo is true, when the whole developing process enter the Composite Process System Designing, the Composite Process Review Design Document will be executed firstly. Specially, when Designers finish the communication with Requirement Analyzers and then promote the first version of Design Document referenced by Final Requirement Document (the Atomic Process Promote Design Document), the Designer should then submit the document to Project Leader (the Atomic Process Submit Design Document)for examining whether this version document is valid or not (maybe Project Leader needs to hold a meeting for discussion) and return a boolean variable Redo (the Atomic Process Check Design Document). If the document reaches a satisfaction criteria (it means the condition Redo is false), the Design Document will be given to Coders by Designer as the norm for system development and coding (the Atomic Process Handover Design Document). On the contrary, if the document does not satisfy a given criteria (the condition Redo is true), it will be returned to Designers with some synthetic suggestions and then be checked again until the document is valid (the Composite Process Review Design Document is executed again and return a new value of Redo).

6 Conclusion

The goal of this paper was to present K-CRIO, an ontology of organizations for their understanding, analysis and also to enable reasoning. The targeted organizations are those dedicated to the realization of products following a given design process. The definition of this ontology relies on OWL and on the concepts used in the CRIO metamodel. The K-CRIO ontology is illustrated by the example of a software development organization following the waterfall process. We are aware that the K-CRIO ontology is not sufficient to provide a Knowledge Management support. It is a first attempt to build a rich ontology for organizations.

Further studies, done with heterogeneous organizations, such as the already done with Scrum [11], will help us in refining K-CRIO and adding new concepts and relationships.

In the meanwhile, the idea is to use K-CRIO as a semantic layer for collaborative softwares in order to enhance Knowledge Management within the targeted organizations. We plan to use our experience of multi-agent systems [15, 16] to design such collaborative softwares References

1. Nattapat Attiratanasunthron and Jittat Fakcharoenphol. A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. Inf. Process. Lett., 105(3):88–92, 2008.

2. Stefano Beco, Barbara Cantalupo, Ludovico Giammarino, Nikolaos Matskanis, and
Mike Surridge. OWL-WS: A workflow ontology for dynamic grid service composition.
In eScience, pages 148–155. IEEE Computer Society, 2005.
3. 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.
4. Ivan Cantador and Pablo Castells. Multilayered semantic social network modelling
by ontology-based user profiles clustering: Application to collaborative filtering.
Springer Verlag Lectures Notes in Artificial Intelligence, pages 334–349, 2006.
5. Massimo Cossentino, Nicolas Gaud, St´ephane Galland, Vincent Hilaire, and Abderrafiaa
Koukam. A holonic metamodel for agent-oriented analysis and design.
In HoloMAS'07, pages 237–246, 2007.
6. Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, St´ephane Galland, and Abderrafiaa
Koukam. aspecs: an agent-oriented software process for engineering
complex systems. Autonomous Agents and Multi-Agent Systems, 20(2):260–304,
march 2010.
7. Mark S. Fox, Mihai Barbuceanu, and Michael Gruninger. An organisation ontology
for enterprise modeling: Preliminary concepts for linking structure and behaviour.
Computers in Industry, 29(1-2):123 – 134, 1996. WET ICE '95.
8. Lars Marius Garshol. Metadata ? thesauri ? taxonomies ? topic maps ! making
sense of it all. Information Science, pages 378–391, february 2004.
9. Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge
sharing. Int. J. Hum.-Comput. Stud., 43:907–928, December 1995.
10. Jomi Fred H¨ubner, Jaime Sim˜ao Sichman, and Olivier Boissier. A model for the
structural, functional, and deontic specification of organizations in multiagent systems.
In Proceedings of the 16th Brazilian Symposium on Artificial Intelligence:
Advances in Artificial Intelligence, SBIA '02, pages 118–128, London, UK, UK,
2002. Springer-Verlag.
11. Yishuai Lin, Vincent Hilaire, Nicolas Gaud, and Abderrafiaa Koukam. Towards an
ontological approach for the description of design processes: the scrum example.
In First International Symposium on Data-Driven Process Discovery and Analysis,
2011. Campione d'Italia (Italy).
12. David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo
Paolucci, Katia Sycara, Deborah L. McGuinness, Evren Sirin, and Naveen Srinivasan.
Bringing semantics to web services with OWL-S. In First International
Workshop on Semantic Web Services and Web Process Composition, pages 243–
277, 2004.
13. Contributor Melonfire. Understanding the pros and cons of the waterfall model of
software development, september 2006.
14. Peter Mika. Ontologies are us: A unified model of social networks and semantics.
J. Web Sem, 5(1):5–15, 2007.
15. D. Monticolo, V. Hilaire, S. Gomes, and A. Koukam. A multi-agent system for
building project memories to facilitate design process. Integrated Computer-Aided
Engineering, 15(1):3–20, 2008.

16. Davy Monticolo, Vincent Hilaire, Abder Koukam, and Samuel Gomes. An egroupware based on multi agents systems for knowledge management. In DEST'07, 2007.

17. Natalya F. Noy and Deborah L. Mcguinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University, 2001.

18. Leo Sauermann, Ludger Van Elst, and Andreas Dengel. A.: Pimo - a framework for representing personal information models. In In Proceedings of I-Semantics 07, JUCS, pages 270–277, 2007.

19. Guus Schreiber. http://www.cs.vu.nl/ guus/public/owl-restrictions/.

20. David Vallet, Phivos Mylonas, Miguel A. Corella, Jos M. Fuentesa, Pablo Castells, and Yannis Avrithis. A semantically-enhanced personalization framework for knowledge-driven media services. pages 11–18, 2005.