Complex Task Ontology Conceptual Modelling: Towards the Development of the Agriculture Operations Task Ontology
Elcio Abrahão, André Riyuiti Hirakawa

To cite this version:

HAL Id: lirmm-03127839
https://hal-lirmm.ccsd.cnrs.fr/lirmm-03127839
Submitted on 1 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives| 4.0 International License
Abstract: Different from domain ontologies, task ontologies must describe the knowledge from its structural and behavioural views, considering aspects as sequence of execution, conditional deviation, external expected and unexpected events interference, pre and post conditions, task granularity, agent participation, geographic localization, resource consumption, production and change. Although the use of conceptual models is well accepted to formally describe domain ontologies, there is little research about conceptual models for complex task ontologies. This paper describes the ongoing research on the Agriculture Operations Task Ontology (AGROPTO) where OntoUML is used to develop conceptual models to describe complex task’s aspects and possible modelling solutions based on Unified Foundation Ontology (UFO). An extension of the E-OntoUML, a language for modelling task ontologies, is suggested to describe methods for modelling task objectives, external event interference, pre/post conditions and task execution state modifications in order to guide future research.

1 INTRODUCTION

The development of knowledge-based systems is directly related to knowledge acquisition, representation, reuse and sharing. In the economy of knowledge (Ducker, 1992) the information is the key point to achieve competitive advantage and develop software tools based on artificial intelligence as deep learning and cognitive computing. The data exponential growth of modern domain applications reinforces the need of information technology tools to: (1) identify and acquire knowledge from different and heterogeneous sources, (2) transform and interpret the data intelligently, (3) provide a common shared formal representation of data and associated knowledge. On domains, like the agriculture field operations, the knowledge-based systems must deal with the big amount of data generated by precision agriculture field sensors, crop cultivation and pre/post harvest procedures. Data should be managed and shared between all stakeholders on the supply chain allowing well-founded strategy decision taken as well as food traceability and wastage control. Ontologies are commonly used to provide a formalism to describe the data and well formed semantics to define concepts associated to the body of knowledge of a given domain as explained in the next section.

1.1 Ontology

According to Gruber (1993), ontology is an explicit specification of a conceptualization and from the knowledge-based systems view, something that “exists” is something that can be formally represented. Guarino (1998) defines different kinds of ontologies according to their level of generality: (1) top-level ontologies, which describe very general concepts such as time, space, objects, actions, events, and are of common use in large domain communities, (2) domain ontologies, which describe the common vocabulary of a domain, (3) task ontologies, which describe generic tasks or activities by specializing terms of the top-level ontology and (4) application ontologies, which describe concepts from a particular domain, by specializing from both domain and task ontologies. There are many works presenting domain ontologies in several computer
science areas as domain engineering, artificial intelligence and semantic web (Guizzardi, 2005). There are also a variety of models proposed to capture the concepts, relations and properties that are relevant for the domain conceptualization through a formal and structured conceptual model (Martins and Falbo, 2008). An important architecture decision to take while developing an ontology is the selection of a foundation ontology as discussed in the next section.

1.2 Foundation Ontologies

Foundation ontology is a high abstract categorization to describe concepts that are commonly used across different domains. Foundation ontologies are systems of philosophically well-founded categories and are domain independent (Guizzardi, Falbo and Guizzardi, 2008). Some foundation ontologies are described in: (Lenat and Guha, 1990; Niles and Pease, 2001; Guizzardi, 2005; Herre et al., 2006). According to Guizzardi (2005), they already prove to be important to increase the quality of modelling languages and conceptual models. The same author proposed a foundation ontology named UFO (Unified Foundational Ontology). UFO is a unification of GFO (Herre et al., 2006), OntoClean (Guarino and Welty, 2009) and DOLCE (Bottazzi and Ferrario, 2009) and it defines things, sets, entities, individuals and types that are the ontological foundation to build consistent and well-formed conceptual models. According to (Guizzardi et al., 2004), UFO is divided into three complementary sets: (1) UFO-A, which defines the core of UFO, (2) UFO-B, which defines the terms related to Perdurants (i.e. type of individual) and (3) UFO-C, which defines the terms related to the spheres of intentional and social things, including linguistic things. Guizzardi et al. (2009) also proposed OntoUML, a conceptual modelling language that contemplates, as modelling primitives, the ontological distinctions proposed by the UFO-A ontology. Automatized computational tools for OntoUML, as described by (Guerson et al., 2015; Moreira et al., 2016) helps in the development of well founded conceptual models and they are used in this research as described in the methodology section.

1.3 Task modelling

Knowledge becomes usable and useful only if it fits the use-context. This is the justification for the expert system technology that relies on heuristic knowledge or on domain experts’ knowledge rather than on objective knowledge like domain theory (Ikeda et al., 1998). Still according to Mizoguchi, Tijerino and Ikeda (1998), expert systems have high performance at the cost of non-reusability of knowledge and low productivity of the knowledge-base development. One of the well-known ideas to solve this problem is the decomposition of expertise into two kinds of knowledge: (1) task-dependent knowledge and (2) domain-dependent knowledge (Mizoguchi, Tijerino and Ikeda, 1995). According to Martins and Falbo (2008), a key factor to capture knowledge is to have a model to represent it. As a model is an abstract representation of a real system, abstraction can be used to remove unnecessary or irrelevant details from the system, so it can be better understood (Gaffar et al., 2004). A model to represent task-dependent knowledge should be able to describe: (1) which tasks are necessary to perform a goal, (2) the agents that perform the task, (3) the task execution time interval and (4) which resources (inputs) are consumed and which products (outputs) are generated (Abrahão and Hirakawa, 2017). A task ontology model can allow agents (humans or machines) to infer knowledge about tasks using semantic techniques for task recognition, negotiation and relocation (Schmidt et al., 2015). Task ontologies will be discussed in more details in the following section.

1.4 Task ontologies

According to Martins and Falbo (2008) and Martins (2009), there are two major kinds of knowledge that should be captured by a task ontology: (1) task decomposition and flow-control, and (2) knowledge roles played by objects from the domain in the task fulfillment. The first kind represents the behaviour view of the task and the second the structural view. Both views need models that could represent the behaviour and structure of a task. UML class and activity diagrams are used to represent task ontology models by (Martins and Falbo, 2008) and where adapted by (Abrahão and Hirakawa, 2017) to describe a task ontology for agriculture operations. Martins (2009) proposed an extension for the OntoUML to describe the knowledge of task ontologies based on events called E-OntoUML. This language uses UML class and activity diagrams to represent task structural and behaviour views. E-OntoUML captures: (1) structural knowledge regarding the roles that domain entities will exert in a task and its relationships, and
(2) behavioural knowledge, which defines the decomposition and flow of control in the subtasks. Although the use of conceptual models is well accepted to formally describe domain ontologies, there is little research about conceptual models for complex task ontologies. As part of an ongoing research, the aim of this paper is: (A) present the initial design of the Agriculture Operations Task Ontology (AGROPTO), a high level generic task ontology independent of crop type; (B) Analyse task ontology structural and behavioural aspects according to E-OntoUML; (C) Propose possible solutions not addressed by E-OntoUML as: (1) task objectives, (2) external event interference, (3) pre and post conditions, (D) Exemplify proposed solutions on an use case for an agriculture field operation of pesticide spraying and (E) Discuss next steps of AGROPTO development and suggest a guide to future work.

2 METHODOLOGY

This section describes the methodologies used to develop the task ontology considering the structural and behavioural aspects, the selection of a foundational ontology, ontology reuse, conceptual modelling and architecture design characteristics.

2.1 Ontology Development Methodology

Although the aim of this work is not to develop a domain ontology for agriculture, it is necessary to define the classes, attributes and relations that will be part of the task ontology structure. It is important to allow the reuse of other domain ontologies (Bontas, Mochol and Tolksdorf, 2005) and the task ontology architecture design should be domain independent and, by itself, reusable by other task and domain ontologies. There are many methodologies recommended to develop domain ontologies as described by (Uschold and King, 1995), (Gruninger, M., and Fox, 1995) and (Falbo, 2004). In this work, the methodology selected was the one proposed by (Noy and Mcguinness, 2000), as showed on Table 1.

### 2.2 Conceptual Modelling

Once a methodology to guide the development of the task ontology was selected, a language to describe the behavioural aspects of the tasks must also be selected. There are some methodologies used to represent the task’s behaviour as described in (Bastos and Ruiz, 2002), (Prata, 2007), (Fersman, Pettersson and Yi, 2002), (Mizoguchi, Tijerino and Ikeda, 1995) and (Rajpathak, Hall and Keynes, 2001). In this work the E-OntoUML proposed by (Martins and Falbo, 2008; Martins, 2009) was selected. E-OntoUML is an extension of OntoUML based on UFO and it provides a concise and robust graphical representation of the structural and behavioural aspects of tasks using, respectively, UML class and activity diagrams. Furthermore, the use of a foundation ontology adds more semantic to task or domain ontologies, making them more concise (Martins, 2009).

### Table 1: Seven steps to create an ontology according to (Noy and Mcguinness, 2000)

<table>
<thead>
<tr>
<th>n#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Determine the domain and scope of the ontology</td>
</tr>
<tr>
<td>2</td>
<td>Consider reusing existing ontologies</td>
</tr>
<tr>
<td>3</td>
<td>Enumerate important terms in the ontology</td>
</tr>
<tr>
<td>4</td>
<td>Define the classes and the class hierarchy</td>
</tr>
<tr>
<td>5</td>
<td>Define the properties of classes—slots</td>
</tr>
<tr>
<td>6</td>
<td>Define the facets of the slots</td>
</tr>
<tr>
<td>7</td>
<td>Create instances</td>
</tr>
</tbody>
</table>

### 2.3 Software Tools

Although it is possible to draw OntoUML diagrams with any regular UML software tool, this work selected an OntoUML editor named Menthor (Moreira et al., 2016) to benefit from the automatized syntax verification and design pattern validation of the tool. To produce the activity diagrams from E-OntoUML, an open source UML software named StarUML (StartUML, 2005) was selected.

3 RESULTS AND DISCUSSION

The main class structures and relations of AGROPTO are presented in this section. The partial results are related to the steps 1 to 4 of the selected development methodology (table 1) and are explained in details in the following subsections.

### 3.1 AGROPTO Initial Design

The conceptual model for AGROPTO reuse the patterns from E-OntoUML to describe the task
Figure 1 - AGROPTO OntoUML model for basic task model structural aspects as task objects, task composition and decomposition, time relations, agent and resource participation and conditional deviation.
execution behaviour view. The E-OntoUML model describes tasks, task aggregation, agents and objects participation in tasks, time relations, task state change and conditional derivation. The E-OntoUML model components (Figure 1) are: (1) Worker: physical agent that executes the task, (2) Activity: a set of 2 or more Agent Actions. (3) Agent Action: represents the task and is specialized in (4) Atomic Agent Action: most granular task that could not be divisible, (5) Complex Agent Action: a composition of 2 or more Atomic Agent Actions and it is specialized in: (6) Interaction: participation of 2 or more Workers to perform a task and (7) Object Agent Interaction: participation of a Worker and a Resource in the task execution. Resources participate in tasks through the relator (8) Resource Participation Action, that specifies the Resource Participation in 4 sub kinds: Resource Creation, Change, Termination and Usage. (9) Resource: a physical object subkind that is specialized in Machinery (vehicle, supplement, tool), Place (geographic localization) and Product (input: needed to perform or used by task or output: generated as result of task execution). Agent Action source and target are framed by a (10) Time Interval specified by time interval realtions derived from Allen’s time relations (Allen, 1983). (11) Situation: is an instant frame of the pre or post states of the task and it satisfies a (12) Preposition Value Specification that represents a conditional derivation on the UML activity diagram. Other classes as (13) Company, (14) Job, and (15) Service Order, where added to the model to describe entities related to the agriculture domain that participates in the structural view of the task ontology. As E-OntoUML did not provide representation for some complex task elements present on agriculture field operations, this paper presents possible solutions in the next subsection.

### 3.2 Complex Task’s Model

Figure 2 shows additional model elements proposed by this work to describe (A) Task Objectives, (B) Pre and Post Execution Conditions, (C) External Event Interference and (D) Task Execution Status Modifications. Task Objective is the goal of the task, i.e. what the task should pursue as a final desirable state. To represent the task objective a specialization of the Situation class called (16) Objective Situation was proposed. As Situation is a snapshot of a state in a moment of time, the Objective Situation is a snapshot of a desirable state where the task goal was achieved. The (17) Condition class describes pre and post execution conditions of the task. Every Condition satisfies one or more (18) Proposition Value Specification, which represents a statement that imposes the Condition that will or not triggers the task execution. One post condition for a task $T_i$ could be a pre condition for the task $T_{i+1}$. Pre and post conditions could then be applied to Agent Action subclasses to an Activity. External event interferences are represented in the model proposed here by the (19) Trigger Event class. The Trigger Event is specialized in two types: (20) Exception Event: unexpected events that may cause an operational interruption that will delay or interrupt the accomplishment of the task objective and happens on uncertain time moment and (21) Ordinary Event: Expected events, but whose exact time and place of occurrence are known only with some degree of uncertainty. The Ordinary Event is specialized in: (22) Planned Ordinary Event: this is a planned and expected event and occurred at a planned and regular time interval, and (23) Unplanned Ordinary Event: Expected event, but occurs at unplanned and irregular time intervals. Trigger Event describes an event that will cause a (24) Condition Modification. The Condition Modification specifies a (25) Condition Action that was specialized in (26) Condition Creation, (27) Condition Change and (28) Condition Termination. A Condition Modification, in turn, changes the attributes of the Condition, and then a Preposition could or could not be satisfied, triggering the execution of a task or causing a modification in the task execution state. The (29) Execution State Action specifies what type of (30) Execution State was modified by the Trigger Event: (31) Interruption, (32) Cancelation or (33) Operational Delay. Both, Trigger Event and Execution State Action happened at a specific point in time (34) Time Point. Figure 3 shows the activity diagram for a use case of pesticide spraying. The activity diagram represents external events that interfere on the task execution (1), pre-conditions for task execution (2) and resource state changes due task execution (3). In this use case the resource tractor can have two pre-conditions to perform the task: be on mechanic working condition and has an operator condition. The resource land has one pre-condition: appropriated soil moisture and the main task, by it self, has one pre-condition too: appropriated wind velocity (on the moment of spraying, to avoid undesired derivation). The external events: mechanic failure
and change of working turn affects the tractor resource. The land resource and main task are affected by the weather external event. The modifications on the task execution state, due to the external event interference, are represented on Figure 4. This figure shows in detail the task state of execution change when an weather event triggers the pre-condition of wind velocity for the main task. The activity node in (1) represents the execution state action of the type interruption. Time points when the event is triggered and the operation need to be interrupted are represented by two instances of the Time Point class (2). Although the E-OntoUML addresses the basic elements of the a task ontology, the proposed solutions presented here goes towards the direction of complement complex task elements descriptions considering well founded ontology development guidelines. An implementation of AGROPTO in the OWL (W3C, 2004) format is available at http://agropto.org. Due to lack of space, E-OntoUML more detailed activity diagrams for AGROPTO where published on the ontology website.

4 CONCLUSION

This work presented possible solutions to modelling complex task ontologies not addressed by the E-OntoUML as: task objectives, pre and post conditions, external event interference and execution state modification. The solutions proposed were used on the Agriculture Operations Task Ontology, a generic task ontology for the agriculture domain in order to describe structural and behavioural aspects of the task execution and generate a well-founded implementation in OWL format. Future research should continue the work on the next steps of the methodology adopted to develop AGROPTO and a more intensive analysis of the solutions proposed here related to the.
concepts of UFO should be done. Well-founded conceptual models for task ontologies will contribute to create more concise and robust knowledge representations on the ontology-engineering domain.

REFERENCES


