Abstract

Semantic annotation of visual models is useful to provide a precise, formal meaning to model elements, thus making them more understandable to people, enabling a deeper analysis of requirements and automated reasoning. We present an approach for the enrichment of visual models of an organization, with annotations characterized by a semantics defined by an organized, structured source of knowledge. In order to carry out the semantic annotation a set of suggestions referring to the use of general and specific ontologies is presented. Moreover, we present a case study to validate the effectiveness of our approach.

© 2013 Published by Elsevier Ltd.

Keywords: semantic annotation, visual models, ontologies, requirements engineering, OntoSem ontology, iStarML.
the same element. Moreover, the amount of information that can be encoded in a human readable label is limited, making visual models difficult to understand outside the organizational context.

The main objective and contribution of this paper is to propose an approach for enriching organizational visual models with annotations, characterized by a semantics encoded in a structured source of knowledge, such as a domain ontology. This enrichment of the organizational models allows us to clarify each element in the model through domain concepts, providing additional support to business analysis during modeling and enabling reuse of information. The proposed approach uses the $i^*$ framework [1], one of the most widespread goal-oriented modeling languages, and the two $i^*$ variants Tropos [2] and service-oriented $i^*$ [3]. However, the proposed approach can be applied to other business modeling techniques.

This paper is structured as follow: Section 2 gives a brief background of the main terms used, Section 3 presents related works, Section 4 describes the proposed approach to enrich organizational models, Section 5 presents a case study to provide a preliminary evidence of its feasibility and performance. Lessons learned are discussed in Section 6 and finally, in Section 7 concluding remarks are given.

2. Background

In this section, we present a brief description of the relevant concepts of the approach: visual modeling and the definition of the term ontology. Finally, the definition of semantic annotation is given.

2.1. Visual Modeling

Visual modeling is the graphic representation of objects and systems of interest using graphical languages. Conceptual modeling through the use of a visual modeling notation is a practice in software development, supporting the communication among stakeholders, involving the development process and the project documentation [4]. Conceptual modeling is concerned with identifying, analyzing and describing the essential concepts of a domain with the help of a modeling language that is based on a small set of basic meta-concepts [5]. Several modeling languages have been proposed for agent-oriented software development [6], goal-oriented analysis [1] and business process modeling [7].

The $i^*$ framework [1] incorporates goal- and agent-oriented modeling and reasoning tools. It is focused on the modeling of intentional and strategic actor relationships. The primitives actor, goal, softgoal, task, and resource are the main concepts of the graphical notation to model an organization.

Tropos [2] is an agent-oriented software engineering methodology that supports the development process of socio-technical systems, from early requirements to software agents implementation, exploiting an $i^*$-based visual modeling language. Service-oriented $i^*$ [3] is the result of revisiting and extending the semantic of the $i^*$ modeling concepts, from a service-oriented perspective. This methodology uses $i^*$ primitives and adds the primitives of service and process.

2.2. Ontologies

Ontologies are considered as key elements for semantic interoperability and to share vocabularies for describing information relevant to a certain area of application. Ontologies are a structured source of knowledge permitting the standardization of concepts, supporting the interoperability at the semantic level [8] and reasoning [9, 10]. According to Guarino [11] the ontologies are classified in different types: Upper level ontologies or General ontologies, Domain ontologies and Application ontologies.

2.2.1. The OntoSem ontology

OntoSem (Ontological Semantics) is a theory of meaning in natural language [12]. The most important feature of OntoSem is to be a practical general ontology.

It is based on a language-independent ontology and a meta-language, which ensures elimination of ambiguity. The ontology is organized as a multiple-inheritance hierarchical collection of frames headed by concepts that are named using language-independent labels. It contains three types of concepts: events, objects and properties. OntoSem contains about 9,000 concepts. This ontology has been successfully applied in [13], [14].
2.3. Semantic annotation

A semantic annotation is a clear, easy to understand, specification that is used to add meaning to a specific object. It enables several advanced analysis and manipulation activities [10]. According to Oren [15], “An annotation is a tuple \((a_s, a_p, a_o, a_c)\), where \(a_s\) is the subject of the annotation, \(a_o\) is the object of the annotation, \(a_p\) is the predicate (the relation between \(a_s\) and \(a_o\)), finally the \(a_c\) is the context in which the annotation is made”. In [15] distinguishes three types of semantic annotation: informal, formal and ontological. An ontological annotation \(A_o\) is a formal annotation \(A_f\), where the predicate \(a_p\) and the context \(a_c\) are an ontological term, and the object \(a_o\) conforms to the ontological definition of \(a_p\).

3. Related Work

The enrichment of organizational models using domain concepts in order to clarify the label of the elements has been poorly explored in requirement engineering. However several approaches apply semantic annotation and ontologies to a) Improve the labeling, b) Discover hidden information, c) Model transformation or d) Resolve the semantic heterogeneity.

In [7], an enrichment of Business Process Models with domain ontologies has been proposed, applying linguistic analysis to process element labels. Domain concepts are used for providing semantic annotation suggestions and to encode the elements into an OWL knowledge base. Our work applies a similar analysis to requirements models. The use of ontologies and semantic annotation to tackle the semantic heterogeneity problem of process models is presented in [9], while [16] presents an application of general ontologies and mapping rules to apply a transformation from Requirement Analysis to System Design.

The Framework SEAN focusses to highly accurate close domains. This framework is based on the manual semantic annotation of documents associated with entities [17]. In [18] the authors describe how the semantic annotation of i* models of actor eco-systems can be used to derive executable process models. In [19] a novel knowledge and information management infrastructure and services for automatic semantic annotation, indexing, and retrieval of documents is described. This approach uses a general ontology and a knowledge base for performing an automatic semantic annotation.

The advantages of using ontologies and semantic annotation have been exploited also in other areas. For instance, in [20] an ontology-based semantic annotation is described, which is embedded in a scenario of a knowledge portal application. Differently with respect to the mentioned approaches, our goal is to clarify the label of the elements represented in the variants i*, Tropos, and service-oriented i* through of semantic annotations. In this way the explicit representation of meanings of elements is promising for more effective sharing and reuse of software requirements and design models. The proposed approach could also improve the process of labelling for visual models and organizational knowledge, making the analyst more aware of the importance of labelling.

4. Enriching the organizational model with semantic annotations

Figure 1 sketches an overview of the proposed approach solution for semantic annotation of organizational models, which consists of two main processes. First, “Semantic annotation suggestion development”, which consists of developing a set of general and specific semantic annotation suggestions. Second, “Extension of iStarML”, which consists of representing the annotated model in iStarML format. The iStarML format could be the input of tools in order to represent the organizational model as ontologies [21], or be useful to integrate the model into a domain ontology at instance level [22]. The final output is an annotated model in the iStarML format.

4.1. Process 1: Semantic annotation suggestion development

This first process consists of three steps (Fig. 1), which guide the annotation process along a set of semantic annotation suggestions, in order to guide the annotation process, as detailed below:

**Step 1.1 Semantic analysis of primitives of i*, Tropos, Service-oriented i*.** We analyze and compare the primitives of each variant of i*: actor, actor type(agent, role, position), goal, softgoal, task, plan, resource,
service and process, with the aim of identifying differences and similarities among them. The result is a single definition for each primitive. Formally, supposing the sets defined as \( \langle V_1, V_2, V_3 \rangle \), where \( V_1 \) represents the first variant to analyze, the \( V_2 \) represents the second variant and \( V_3 \) represents the third variant. Given the following domain elements \( \langle p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11} \rangle \), where \( p_1 \) represents the actor primitive, \( p_2 \) represents the type actor “agent”, \( p_3 \) represents the type actor “role”, \( p_4 \) represents the type “position”, \( p_5 \) represents the goal primitive, \( p_6 \) represents the softgoal primitive, \( p_7 \) represents the task primitive, \( p_8 \) represents the plan primitive, \( p_9 \) represents the resource primitive, \( p_{10} \) represents the service primitive and \( p_{11} \) represents the process primitive.

We define each set with the respective domain elements, e.g. \( V_1=\{p_{1,1}, p_{2,1}, p_{3,1}, p_{4,1}, p_{5,1}, p_{6,1}, p_{7,1}, p_{9,1} \} \), \( V_2=\{p_{1,2}, p_{2,2}, p_{3,2}, p_{4,2}, p_{5,2}, p_{6,2}, p_{7,2}, p_{9,2} \} \), \( V_3=\{p_{1,3}, p_{5,3}, p_{6,3}, p_{7,3}, p_{9,3}, p_{10,3}, p_{11,3} \} \). In this case, \( V_1 \) represents \( i^* \), \( V_2 \) represents Tropos and \( V_3 \) service-oriented \( i^* \), with the respective primitives. The process consists of analyzing \( p_{1,1} \) of the set \( V_1 \), \( p_{1,2} \) of the set \( V_2 \) and \( p_{1,3} \) of the set \( V_3 \), aiming at identifying the differences and similarities between them. The result is to obtain a single definition for each primitive that integrates similar features among variants. So, we obtain \( \{D_1, D_2, D_3, D_4, D_5, \ldots D_{11} \} \), where \( D_n \) represents the integrated definition of each primitive: \( D_1 \) the primitive actor, \( D_2 \) the actor type “agent” and so on.

**Step 1.2 Analysis of general and domain ontologies.** This step consists of analyzing the hierarchy and relationships between concepts of general and domain ontologies, to establish relationships between the primitives recognized in Step 1, towards one or more concepts.

Supposing the concepts \( C_1 \) and \( C_2 \) are compared with the definition \( D_1 \), if \( C_1 \) and \( C_2 \) help to describe or defined a \( D_1 \), then all the instances of the primitive \( D_1 \) should be mapped with \( C_1 \) and \( C_2 \). In a general way, if \( C_n \) concept describes a \( D_n \) definition, so all the instances \( i_1, i_2, i_3, \ldots i_n \) of \( D_n \) should be mapped to \( C_n \).

**Step 1.3 Development of semantic annotation suggestions.** This step consists of establishing formally each primitive into one or more domain concepts. The result of this step is a set of general semantic annotation suggestions and a set of specific semantic annotation suggestions [22]. The first suggestions can be applied to any domain ontology. The second can be applied to OntoSem ontology and to its extensions.

The general suggestions have certain freedom to relate each primitive with domain concepts. For example, the primitive “goal” should be mapped into domain concepts that describe a clear and precise condition, interest or desire (Table 1).

While, the specific semantic annotation suggestions present the relationships of each primitive with one or more domain concepts of OntoSem ontology. For example, the primitive “goal” \( (D_3) \) should be mapped into one of the super-concepts “mental-event \( (C_1) \)”, social-event \( (C_2) \) and mental-object \( (C_3) \)” (Table 2). This means that all the instances \( (i_n) \) of a primitive of type goal should map into one of the concepts \( \{C_1, C_2, C_3, \ldots \} \).
independently of the model domain. The general semantic annotation suggestions are presented in Table 1 and the specific semantic suggestions are presented in axiomatic form in Table 2.

Table 1. General semantic annotation suggestions

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>An actor (including the actor types) should be mapped into domain concepts</td>
</tr>
<tr>
<td></td>
<td>that describe an organization, agent, tangible entity, or intangible entity.</td>
</tr>
<tr>
<td>Goal</td>
<td>A goal should be mapped into domain concepts that describe a clear and precise</td>
</tr>
<tr>
<td></td>
<td>condition, interest or desire.</td>
</tr>
<tr>
<td>Softgoal</td>
<td>A softgoal should be mapped into domain concepts that describe an interest or</td>
</tr>
<tr>
<td></td>
<td>desires not clear-cut satisfaction criteria.</td>
</tr>
<tr>
<td>Plan, Task and Process</td>
<td>A plan, a task and a process should be mapped into domain concepts that de-</td>
</tr>
<tr>
<td></td>
<td>scribe a concrete action or activity.</td>
</tr>
<tr>
<td>Resource</td>
<td>A resource should be mapped into domain concepts that represent a physical</td>
</tr>
<tr>
<td></td>
<td>object or informational entity.</td>
</tr>
<tr>
<td>Service</td>
<td>A service should be mapped into domain concepts that represents a function-</td>
</tr>
<tr>
<td></td>
<td>ality or specification of services.</td>
</tr>
</tbody>
</table>

Table 2. Specific semantic annotation suggestions between model elements (ME) and OntoSem super-concepts (SC)

<table>
<thead>
<tr>
<th>Merging axioms</th>
<th>Intuitive meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME : Actor → SC : object</td>
<td>A model element of type actor can be annotated only with (can represent only) the super-concept object.</td>
</tr>
<tr>
<td>ME : Goal → SC : mental − event ∧ SC : social − event ∧ SC : mental − object</td>
<td>A model element of type goal can be annotated only with (can represent only) the super-concepts mental-event, social-event and mental-object.</td>
</tr>
<tr>
<td>ME : SoftGoal → SC : abstract − object</td>
<td>A model element of type softgoal can be annotated only with (can represent only) the super-concept abstract-object.</td>
</tr>
<tr>
<td>ME : Task ∧ Plan ∧ Process → SC : active − cognitive − event ∧ SC : social − event ∧ SC : physical − event</td>
<td>A model element of type task, plan and process can be annotated only with (can represent only) the super-concepts active-cognitive-event, social-event and physical-event.</td>
</tr>
<tr>
<td>ME : Resource → SC : physical − object ∧ SC : mental − object</td>
<td>A model element of type resource can be annotated only with (can represent only) the super-concepts physical-object and mental-object.</td>
</tr>
<tr>
<td>ME : Service → SC : social − event</td>
<td>A model element of type service can be annotated only with (can represent only) the super-concept social-event.</td>
</tr>
</tbody>
</table>

4.2. Process 2: Extension of the iStarML

In three steps, as displayed in Figure 1, we achieve a representation of the annotated model in iStarML format.

Step 2.1 Analysis of iStarML format. We analyse the iStarML XML-based model interchange format [23] and the ccistarml¹ Java package for the i* mark-up language which represents an i* model in iStarML format, to identify the XML attributes possible that could store the semantic annotation for each element of the model.

Step 2.2 Extension of iStarML format. We extend the iStarML format adding the XML attribute “sandbox”, which allows us to identify each element of the model with the respective domain concepts. Supposing a goal element “Get credit info” and the semantic annotations are “counseling” and “information-obtain”. This element represented in iStarML is denoted, such as: <name=“Get credit info” type=“goal” sandbox=“counseling information-obtain”>.

Step 2.3 Extension of iStarML plug-in to JUCMNav. We automate the generation of the annotated model represented in iStarML format extending the JUCMNav tool². The JUCMNav “import-export plug-

¹see http://www.upc.edu/gessi/istarml/resources.html for details.
²JUCMNav, developed by D. Amyot et al. at the University of Ottawa, is a graphical editor and an analysis and transformation tool for the User Requirements Notation, consisting of i* goal models and use case maps. Available at http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome.
We extend the graphical editor in order to add the semantic annotation. In particular, the “description” property is used, together with a demarking symbol “@”. We extend the plug-in also to generate the new iStarML file adding the semantic annotation using the new attribute “sannotation”, resulting in an automatic generation of iStarML files with model annotations.

5. Case study

In order to validate our approach, we used it on several case studies. In this paper we describe each phase of our approach using the smart card-based payment system case study, which has been modelled in [24]. A cardholder depends on a card issuer to be allocated a smart card, for the terminal depends on him to present his card for each transaction. The card issuer in turn depends on the card manufacturer and software manufacturer to provide cards, devices, and software. The data owner is the one who has control of the data within the card. He depends on the terminal to submit transaction information to the central database. A fragment of the i* Strategy rationale model for the case study is shown figure 2.

![Figure 2: A fragment of i* strategic rationale model [24]](image)

Our approach of enriching organizational models through semantic annotation consists of two main processes. First process, the “Annotation process”, which consist of taking the set of semantic annotation suggestions developed in step 1.3 of our approach solution proposed (fig. 1). The steps 1.1 and 1.2 are carried out only to develop the set of semantic suggestions. The suggestions are the guideline for the annotation process. Second process, “Export to iStarML format”, consists in generating the iStarML format from an annotated model using our plug-in extended (step 2.3 fig. 1). In the same case, steps 2.1 and 2.2 are carried out only to automate the generation of iStarML.

**Process 1: Annotation process.** We annotate the model following the specific suggestions in Table 2. For example, in the case of the element of type goal the suggestion indicates that $\text{ME} : \text{Goal} \rightarrow \text{SC} : \text{mental-event} \land \text{SC} : \text{social-event} \land \text{SC} : \text{mental-object}$. This axiom defines that an element of type goal can be annotated with the superconcepts: mental-event, social-event and mental-object. The proceeding is going in-depth of these superconcepts in OntoSem and to find-out the more appropriate domain concept for the element of type goal. The concepts selected should be congruent with the description of the element. For example, figure 2 shows the goal “New Account”, following the specific suggestions the concept “open-account” from “social-event” superconcept means “The event of opening a bank account”. Other concept related is “bank-account” from “mental-object” describes “a bank account in the short-term investment market, usually requiring a minimum deposit”. Both concepts added additional information to the label of the element. In Figure 3 on the left side a fragment of annotated model is shown.

We propose that a domain concept can represent different instances of a model. For instances, the goal “New Account” and the task “Create New Account”. In this case, two different model elements are integrated with the same domain concepts “@open-account” and “@bank-account” (Figure 3). Moreover,
the categorizing of the elements of the model into domain concepts allows the reuse of parts of the elements when creating new models, the detection of cross-item relationships and reasoning between elements.

**Process 2: Export to iStarML format.** This process consists of automating the representation of annotated model to iStarML format, using the extended plug-in for JUCMNav. In the right-side of Figure 3 a fragment of the annotated model represented in iStarML is shown.

![A fragment of annotated i* strategic rationale model](image)

**Fig. 3. A fragment of annotated i* strategic rationale model**

6. **Lessons Learned**

This approach addresses the problem of inconsistency in the labeling of the elements of an organizational model. We emphasize the use of semantic annotation and ontologies to tackle this problem. We present a set of semantic annotation suggestions to guide the annotation process and the extension of iStarML format. The results of the case study lead us to the following observations:

i) In the case study we annotated 9 goals, 3 softgoals, 28 tasks, 2 resources, 3 goal dependencies, 6 softgoal dependencies, 2 task dependencies and 10 resource dependencies. The annotation process is time-consuming due to being a manual process. We envision that a semi-automated annotation process, which provides suggestions, can improve performance.

ii) Based on our experience, the expressiveness of OntoSem ensures elimination of ambiguity of labels of model elements and is able to capture a detailed and precise meaning.

iii) The representation of the annotated model in iStarML format could be useful to integrate the model into a domain ontology at instance level.

iv) Enriching the models allows us: to clarify the elements of the model and their description, to permit the interoperability among i* variants through of domain concepts, and to be able to find and reuse of parts of a model when creating new models.

7. **Conclusion and future work**

In this paper, we presented an approach for enriching organizational models with annotations characterized by a semantics explicitly organized in a structured source of knowledge. These annotations clarify the label of a model element by associating it to domain concepts. The standardization of elements through ontological concepts improves the labeling activity, the process of analysis and the reuse of information. Moreover, a model annotated with semantic annotations is clear for humans and accessible to machines because of the integration with ontologies.

We described a set of specific and general semantic annotation suggestions to annotate the elements of the models represented in the i* variants, as Tropos and service-oriented i*. The extension of the iStarML format and of an existing plug-in for the jUCMNav modeler allows us to automatically generate the annotated model represented in iStarML.
The case study presented provides preliminary evidence about the effectiveness of our annotation process. We are currently working on a tool to integrate an organizational model into an ontology together with its semantic annotation. Moreover, for future work we propose to automate the process of suggestions for each element of a model, using techniques of natural language processing. Also, an empirical study with subjects could bring evidence of the effectiveness of semantic annotation for the comprehension of labels in conceptual models.

References