An Ontology-Based Methodology for Integrating *i** Variants

Karen Najera[1,2], Alicia Martinez[2], Anna Perini[3], and Hugo Estrada[1,2]

[1] Fund of Information and Documentation for the Industry, Mexico D.F, Mexico
{karen.najera, hugo.estrada}@infotec.com.mx
[2] National Center of Research and Technological Development, Cuernavaca, Mexico
amartinez@cenidet.edu.mx
[3] Bruno Kessler Foundation - IRST, Center for Information Technology, Trento, Italy
perini@fbk.eu

Abstract. Many variants of the *i** Framework have been developed since its definition. For instance, *i**-based modeling languages have been proposed for agent-oriented and service-oriented software development, for modeling risk, security requirements, and so on. The integration of models expressed in *i** variants poses interoperability problems. This issue has been faced at different levels, e.g. through unified metamodels, or with an interchange format to depict *i** models. In our research work, we propose a practical approach to tackle the interoperability problems, exploiting ontology-based techniques. In a previous work, we presented an ontological representation of the *i** metamodel and a tool to automatically transform an *i** model to an instance of this ontology. In this position paper, we describe a methodology to integrate *i** variants through the ontological representation of the *i** metamodel, and a mechanism to support the understanding of models expressed in those variants. As example of application, we present the integration of *i**, Tropos and Service-oriented *i** by using ontologies and a tool to automatically transform models expressed with those variants in terms of that ontology.

Keywords: *i**, organizational modeling, ontologies, model-driven engineering, model transformations.

1 Introduction

The *i** Framework [9] is a widely used organizational modeling technique. Since its definition, many research projects have used it in different application domains, hence many *i** variants have been proposed. For instance, Tropos [5] for agent-oriented development, Service-oriented *i** [3], and many others. Commonly, differences among variants consist of the addition of constructs or changes in the semantics of the existing ones in the original framework, as remarked also in [4], which refers to a study of 63 papers on *i** related methodologies, which were published between 2006-2010. Regardless of the degree of difference in variants, the integration and understanding of models expressed in *i** variants poses interoperability problems. This issue has been faced at different levels, e.g. through unified metamodels ([1, 6]), and with an interchange format to depict *i** models [2]. In this research work [7], we aim to show that the use of ontologies

can provide a practical approach towards tackling the *i** variants interoperability problem. As a preliminary stage, we have proposed the use of ontologies to integrate *i** variants, propitiating the understanding of the variants and their models by means of a common language established by the ontologies. We used the Web Ontology Language "OWL" to describe ontologies, since it is a standard of the Semantic Web which facilitates greater machine interpretability than XML or RDF, and provides a formal semantics. With our proposal, we bring the advantages of ontologies to the organizational modeling domain, such as querying, reasoning and organizational data specified in a Semantic Web format. Our initial results have been presented in [8], namely: a) The development of an ontology-based metamodel for *i** called OntoiStar and b) a tool-supported process to generate ontologies from models expressed in *i**. In this paper, we present: a) a methodology to integrate *i** variants by using ontologies on the basis of OntoiStar, b) an example of application integrating the variants *i**, Tropos and Service-oriented *i**, and c) the extension of our tool-supported process to support the understanding of models expressed with the integrated *i** variants.

2 Objectives

The main objective of our research work [7] is *The integration of i* variants through the use of an ontology and the automatic representation of models expressed with those variants in terms of the ontology with* i* *variants integrated. Thereby supporting the understanding of variants and their models.* For the fulfillment of this objective four specific objectives have been identified:

1. The development of an ontology called OntoiStar for representing the core constructs of the *i** variants.
2. The development of a methodology for integrating the constructs of several *i** variants through the use of ontologies the basis of the ontology OntoiStar.
3. The application of the methodology to different i* variants, generating an ontology with *i** variants integrated.
4. The automatic transformation of a model represented with one of the integrated *i** variants into an ontology derived from the concepts of the ontology with the *i** variants integrated.

The first specific objective has been addressed in [8]. In this paper we focus on the last three specific objectives.

3 Scientific contributions

Our scientific contributions are related to the accomplishment of the specific objectives 2, 3 and 4 of our research work. Therefore, in this section we present our proposed methodology, which describes how to use ontologies to achieve the integration of *i** variants. Moreover, we present the application of the methodology to *i**, Tropos and Service-oriented *i**.

3.1 Integration methodology

Our proposed integration methodology provides the guidelines to integrate constructs of several *i** variants into an ontology which extends OntoiStar [7, 8].

The ontology with i^* variant integrated has been named OntoiStar+ in a general way, to indicate this ontology considers the constructs of two or more i^* variants, no matter which or how many they are. The integration methodology to obtain OntoiStar+ is based on Model-Driven Engineering (MDE), since it is generated on the basis of OntoiStar which was developed through MDE at the level of metamodels. The methodology consists of two phases: 1) the development of an ontology for each i^* variant desired to integrate and 2) the integration of the ontologies of the i^* variants generating the ontology OntoiStar+.

Phase 1) Development of an ontology for a specific *i variant**
In this phase the ontology for a specific i^* variant is generated. The resultant ontology is based on OntoiStar. Therefore, additional constructs of a specific i^* variant are added into OntoiStar. This phase may be performed several times when more i^* variants need to be integrated. It consist of four steps:

I. Identify. The first step corresponds with the identification of the additional constructs of the i^* variant which are not part of the ontology OntoiStar.

II. Categorize. The second step corresponds to the categorization of the additional constructs identified in the first step. Four categories allied with the elements of metamodels have been defined. A construct is categorized as:
Concept, when it corresponds to a representation of something in the real world.
Relationship, when it corresponds to a relationship of two or more concepts.
Attribute, when it is used to define a property or characteristic of a concept. It may also refer to or set the specific value for a given instance of such.
Attribute value, when it corresponds to those values that belong to an additional construct which has been categorized as attribute.

III. Transform. The third step corresponds to the transformation of the additional constructs into ontological constructs, i.e. classes, properties and axioms in OWL. Two sets of transformation rules have been proposed according with the i^* variant specification. One for those constructs identified from a metamodel described in the UML language, and other for constructs identified from a textual description (Table 1).

IV. Classify. In [7, 8] we presented the OntoiStar taxonomy which was defined according with the core i^* concepts specified in [2]. This step corresponds with the classification, within OntoiStar taxonomy, of the new OWL classes resulting from the third step. Therefore, a new OWL class is subclass of the class:
Actor, if the OWL class describes a new type of actor.
Actor Relationship, if the OWL class describes a new type of actor relationship.
Dependency, if the OWL class describes a new dependency relationship. The dependency basic structure has been already defined in OntoiStar.
Boundary, if the OWL class describes a new type of boundary.
Intentional Element, if the OWL class describes a new type of intentional element.
Intentional Element Relationship, if the OWL class describes a new type of intentional element relationship.

Table 1. Transformation rules

From metamodel	From textual description
Each concept, concept relationship and enumeration class is represented as a class in OWL.	Each concept, concept relationship and attribute is represented as a class in OWL.
Each association is represented as an object property in OWL.	If a class in OWL is created due a concept relationship, two object properties are created to complete the relationship.
Each class property is represented as axioms in OWL.	-
Each enumeration element is represented as a class instance of the owner enumeration class in OWL.	If a class in OWL is created due an attribute, each attribute value is represented as a class instance of the corresponding attribute class in OWL.
Attributes. Each enumeration type is represented as an object property in OWL. Each primitive data type is represented as a data property in OWL.	If a class in OWL is created due an attribute, an object property is created to complete the representation of the attribute.

Phase 2) Integration of the ontologies of the *i variants**
In this phase, the ontologies of the i^* variants (generated in phase 1) are integrated by means of an iterative merging process, obtaining as a result, the ontology OntoiStar+. The merging process consist of applying a merging function to the i^* variant ontologies, two at a time, till obtain the ontology with all the desired i^* variants. It is applied first to two i^* variant ontologies, then, the resultant ontology is merged with another i^* variant ontology, and so on. See for instance Fig. 1. The merging function consist of bringing together all the constructs of two ontologies, taking into account that duplicated constructs are only considered one time in the final merged ontology. The merging function has been implemented in the tool described in 3.3. With this approach a user can select the ontologies to merge according to the i^* variants the user works with.

3.2 Application of the integration methodology

We have applied the integration methodology to i^*, Tropos and Service-oriented i^*. The integration results are presented in Table 2 (attributes were omitted due to space). The first two columns describe constructs already included into OntoiStar. The next three columns contain the additional constructs of each variant (N.A. indicates there is no additional constructs). First, we performed Phase 1, therefore, additional constructs of each variant were identified, categorized, transformed and classified in order to obtain the ontology for each variant. For instance, in Service-oriented i^*, the construct "Service" was identified, then, it was categorized as concept and transformed as an OWL class. Finally, it was classified as an Intentional Element, therefore, placed as a sub class of the Intentional Element class in the OntoiStar taxonomy. Then, Phase 2 was carried out by merging the ontology for i^*, the ontology for Tropos and the ontology for Service-oriented i^*, obtaining as a result the ontology OntoiStar+ with the three variants integrated. The integration process is represented in Fig. 1. The

Table 2. Additional constructs of each *i** variant.

OntoiStar		***i****	**Tropos**	**S-O *i****	**OntoiStar+**
***i** concept**	**Types**	**Types**	**Types**	**Types**	**Types**
Actor	Agent, Role, Position	N.A.	N.A.	N.A.	Agent, Role, Position
Actor Relationships	Is_part_of, Is_a, Plays, Covers, Occupies	Instance_of	N.A.	Subordination	Is_part_of, Is_a, Plays, Covers, Occupies, Instance_of, Subordination
Dependency	Goal, Softgoal, Task, Resource	N.A.	Plan	Service, Process	Goal, Softgoal, Task, Resource, Plan, Service, Process
Boundary	-	N.A.	N.A.	N.A.	-
Intentional element	Goal, Softgoal, Task, Resource	N.A.	Plan	Service, Process	Goal, Softgoal, Task, Resource, Plan, Service, Process
Intentional element relationship	Contribution, Decomposition, MeansEnd	N.A.	N.A.	Service, Service-goal, Process	Contribution, Decomposition, MeansEnd, Service, Service-goal, Process

last column of Table 2 contains the constructs included in the resultant ontology OntoiStar+.

3.3 Automatic transformation from *i** based models to ontologies

In this section, we present TAGOOn (Tool for the Automatic Generation of Organizational Ontologies), an extension of the one in [8]. TAGOOn provides the basis to support the automatic transformation from models expressed with different *i** variants in ontologies. The current version supports models expressed with *i**, Tropos and Service-oriented *i**. The transformation process starts with the representation of models using iStarML [2]. Some additional constructs of the supported variants are not defined in its grammar. Therefore, we have used the open options of the iStarML specification to represent them. For instance, using the attribute 'type', concepts as plan, service and process could be defined in the tag $< ielement >$; and relationships as service relationship and service dependency could be defined using in the tag $< ielementLink >$.

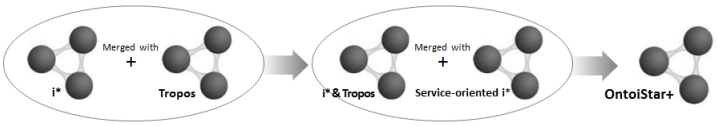

Fig. 1. *i**, Tropos and Service-oriented *i** integration process

In order to perform the automatic transformation, TAGOOn parses the iStarML file and, according to pre-defined mapping rules, it instantiates the corresponding classes and properties in the ontology OntoiStar+. The output of the tool corresponds to the ontology OntoiStar+ with instances that represent the knowledge depicted in the organizational model. It shapes an organizational knowledge base in which is possible to apply services offered by ontologies such as querying and reasoning. Furthermore, it can be edited with an ontology editor, or it can be the input of development or reasoning platforms supported by ontologies.

4 Conclusion and Future Work

In this paper, we presented a further step in the achievement of interoperability of *i** variants, which extends a previous work were an ontological representation of the *i** metamodel was proposed [8]. Specifically, we presented a methodology to integrate several *i** variants into an ontology and our supporting tool TAGOOn. With our proposal, we bring advantages of ontologies such as querying and reasoning, to the organizational modeling domain. Moreover, as the organizational knowledge is represented in OWL, it could be available to be exploited and consumed in the Semantic Web by paradigms such as Linked Data.

The main steps we intend to address in our future work can be summarized as follow: a) To take into account the semantic of the *i** variants during the creation of ontologies; b) To propose inference rules that enable the redefinition and adjustment of *i** based models according with the semantic and differences of *i** variants; c) To extend TAGOOn to carry out the automatic transformation of *i** based models from one *i** variant to other *i** variant.

References

1. C. Cares, X. Franch, E. Mayol, and C. Quer. A Reference Model for i*. In *Social Modeling for Requirements Engineering*, pages 573–606. MIT Press, 2010.
2. C. Cares, X. Franch, A. Perini, and A. Susi. Towards interoperability of i* models using istarml. *Computer Standards & Interfaces*, 33(1):69–79, 2011.
3. H. Estrada. *A service-oriented approach for the i* framework*. PhD thesis, Valencia University of Technology, Valencia University of Technology, Valencia, Spain, 2008.
4. X. Franch. The i* framework: The way ahead. In *Sixth International Conference on Research Challenges in Information Science RCIS'12*, pages 1–3, 2012.
5. P. Giorgini, J. Mylopoulos, A. Perini, and A. Susi. The Tropos Methodology and Software Development Environment. In *Social Modeling for Requirements Engineering*, pages 405–423. MIT Press, 2010.
6. M. Lucena, E. Santos, C. T. L. L. Silva, F. M. R. Alencar, M. J. Silva, and J. Castro. Towards a unified metamodel for i*. In *Research Challenges in Information Science*, pages 237–246, 2008.
7. K. Najera. An ontology-based approach for integrating i* variants. Master's thesis, National Center of Research and Technological Development, Cuernavaca, Morelos, Mexico, 2011. `www.tagoon.semanticbuilder.com/NajeraThesis.pdf`.
8. K. Najera, A. Perini, A. Martínez, and H. Estrada. Supporting *i** model integration through an ontology-based approach. In *iStar*, pages 43–48, 2011.
9. E. S.-K. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, University of Toronto, Toronto, Ont., Canada, 1996.