Towards interoperability of *i** models using iStarML

Carlos Cares[1,2] ccares@essi.upc.edu
Xavier Franch[1] franch@essi.upc.edu
Anna Perini[3] perini@itc.it
Angelo Susi[3] susi@itc.it

[1] Universitat Politècnica de Catalunya, C/Jordi Girona, 1-3, 08034 Barcelona, Spain

[2] Universidad de la Frontera, Avenida Francisco Salazar 01145, Temuco, Chile

[3] ITC-irst, Trentine Culture Institute, Scientific and Technological Research Centre, 38050 Povo, Trento, Italy

January 2010

Towards interoperability of *i** models using iStarML

Carlos Cares [a,b,*], Xavier Franch [a], Anna Perini [c], Angelo Susi[c]

[a] *Technical University of Catalonia, C/Jordi Girona, 1-3, 08034 Barcelona, Spain*

[b] *University of La Frontera, Av. Francisco Salazar 01145, Casilla 54-D, Temuco, Chile*

[c] *FBK-irst, Via Sommarive 18, I-38050, Povo, Trento, Italy*

Abstract

Goal-oriented and agent-oriented modelling provides an effective approach to the understanding of distributed information systems that need to operate in open, heterogeneous and evolving environments. Frameworks, firstly introduced more than ten years ago, have been extended along language variants, analysis methods and CASE tools, posing language semantics and tool interoperability issues. Among them, the *i** framework is one the most widespread. We focus on *i**-based modelling languages and tools and on the problem of supporting model exchange between them. In this paper, we introduce the *i** interoperability problem and derive an XML interchange format, called iStarML, as a practical solution to this problem. We first discuss the main requirements for its definition, then we characterise the core concepts of *i** and we detail the tags and options of the interchange format. We complete the presentation of iStarML showing some possible applications. Finally, a survey on the *i** community perception about iStarML is included for assessment purposes.

Keywords: agent orientation; requirements engineering; *i**; interoperability

1. Introduction

Requirement Engineering (RE) has been defined as the branch of software engineering (SE) concerned with the real-world goals for functions of, and constraints on, software systems [1]. RE is inherently broad, interdisciplinary and open-ended because it embraces from real life situations to mathematical specification languages.

Goal-oriented RE methodologies were introduced more than ten years ago. They have been recognized to play a crucial role to model the domain and to identify the requirements of a new software system, through the understanding of stakeholders' domain goals and of their strategic dependencies for goal achievement [2, 3]. Goal-oriented approaches have been formulated either as formal frameworks, e.g. KAOS [4], or as rigorous (but not formal) ones, both of them proposing their own modelling language, with a specific set of conceptual entities, a graphical notation to depict models, and a set of analysis techniques. Among rigorous frameworks, the *i** (pronounced *eye-star*) framework [5] is one of the most, if not the most, widespread and adopted by the

———

RE community. Several indicators support this statement, from the increasing number of scientific papers and experience reports on *i** presented in RE&SE journals and papers; the periodic organization of an *i** workshop; the construction of the *i** wiki (http://istar.rwth-aachen.de/tiki-index.php) with more than 35 universities and organizations currently registered; the imminent publication of a monograph on *i** [6]; the recognition of an *i**-based language like URN as a telecommunication standard (standard Z.150); and the offering of tutorials in world-leading conferences like IEEE RE (2008).

The *i** framework provides the ability to model concepts such as actors, roles and agents, and to reason about them. Also relevant in this kind of modelling is the assignment of goals to actors or agents, which has been the base of agent-oriented software methodologies [7]. Combining goals and agents altogether allows labelling the *i** framework both as agent-oriented and goal-oriented. *i** is one of the most widespread modelling languages by itself and also as part of the Tropos SE methodology [8].

As a side effect of this growing interest around *i**, several extensions to the original framework have been defined; a summary can be found in [9]. They have been applied not only to RE but also to other fields as organizational patterns [10], agent networks simulation [11], and agent security patterns [12] among others. These more recent approaches aim at dealing with the increasing complexities in developing nowadays software-intensive systems, which need to operate in open, heterogeneous and evolving environments. Also, several *i**-related tools have been built. They offer capabilities for editing *i** models, for analyzing them and for applying techniques over them.

Therefore, problems on: (i) the consistency of the semantics of the different language variants, and on (ii) how to exploit reasoning services offered by the tools each one requesting specific modelling formats, have become relevant.

The first of these two problems has been recently addressed by different approaches. For instance, in [13] the authors propose to root the modelling language to a domain-independent metamodel that takes into account more basic entities motivated by philosophical cognitive science theories (e.g., object, event). These ideas have been illustrated with respect to the concept of goal. In [14], the problem of language variants is analyzed along a set of ten concept properties like reflexivity, boundary, symmetry, with the aim to refine the syntax of *i**-based modelling languages. In [9], a comparison of some variants of *i** is presented and a metamodel for embracing the commonalities is proposed; customization on the source variants is projected by using refactoring techniques. In [15] the Tropos metamodel includes the *i** constructs and its use has promoted both extensions to Tropos such as Secure Tropos [16] and tool implementations such as TAOM4E [17].

The diversification of *i** applications has produced semantic variations which have implied practical problems concerning interoperability of tools, e.g. goal-analysis tools, modelling tools, metric calculation tools, etc., because each tool works over a particular *i** variant. This prevents sharing models among tools or combining two of them for providing enhanced functionalities. This problem could be unsolvable if variants were radically different but as shown in [9] this is not the case: changes are either minor differences on basic *i** constructs. For this reason, the objective of overcoming that interoperability limitation seems feasible.

In our work we focus on practical issues related to the *i** tools interoperability problem. In particular, our main objective is to provide a representation where differences and similarities among *i** variants are explicit, generating a common representational framework for the *i** community and, in spite of the differences, enabling effective communication inside the community, tool interoperability and a common representation for repository of *i** models. Our proposal is based upon the definition of an XML interchange format for *i** diagrams, called iStarML. iStarML includes six basic categories of core concepts, transformed into six abstract core concepts, common to all of *i**-based modelling languages. Using XML as basic infrastructure makes it possible to have other goals, specifically to take advantage of the XML format for Internet communication and also to use general-purpose XML tools.

The rest of the paper is structured as follows. In Section 2 we show the basic features of the *i**-based framework, discuss about *i** variations and present the related interoperability problem. Next, in Section 3, we outline a set of requirements for an interchange format showing the analysis that generated the essential principles of the language design. Also in this section we explain how we have considered both stable *i** concepts and variant *i** concepts. In Section 4, we illustrate the facilities of using iStarML showing four generic application scenarios by pointing to particular technological examples. Finally, in Section 5, we show the results of a survey applied to representative members of the *i** research community which have expressed their perception about the tool interoperability problem in the *i** community, and about how iStarML could solve this problem. Conclusions include an overview of benefits and drawbacks of the proposal, related work comments, mainly focused on XMI and MDA proposals and the planned future work.

2. The *i** framework and its variations

In this section we briefly introduce the *i** framework and the main features of its variants, highlighting differences and similarities, with the aim of characterizing the interoperability problem in this framework. We generalize then this discussion proposing a formal definition of this interoperability problem and of the solution we propose.

2.1. The i framework and its variations*

The *i** framework [5] was formulated for representing, modelling and reasoning about socio-technical systems (e.g. [18, 19]). Its modelling language is constituted basically by a set of graphic constructs which can be used in two models. Firstly, the Strategic Dependency (SD) model, which allows the representation of organizational *actors*, specialized on *roles*, *positions* and *agents* (positions *cover* roles; agents are physical *instances*). Actors can be related by *is-a*, *is-part-of*, *covers*, *instance-of*, *plays* and *occupies* relationships. Also actors can have social dependencies. A *dependency* is a relationship among two actors, one of them, named *depender*, who depends for the accomplishment of some internal intention from a second actor, named dependee. The dependency is then characterized by an intentional element (*dependum*) which represents the dependency's element. The primary intentional elements are: *resource*, *task*, *goal* and *softgoal*. A *softgoal* represents a goal that can be partially satisfied, or a goal that requires additional agreement about how it is satisfied. They have usually been used for representing non-functional requirements and quality concerns.

Secondly, the Strategic Rationale (SR) model represents the internal actors' rationale. The separation between the external and internal actor's worlds is represented by the actor's boundary. Inside this boundary the rationality of each actor is represented using the same types of intentional elements described above. Additionally these intentional elements can be interrelated by using relationships such as *means-end* (e.g., a task can be a mean to achieve a goal), *contributions* (e.g., some resource could contribute to reach a quality concern or softgoal) and *decompositions* (e.g., a task can be divided into subtasks). In Figure 1 we show an excerpt of an *i** model for an academic tutoring system. There appear most of constructs already described. The intuitive meaning of this context should help to capture the practical use and the semantics of the *i** framework. For a through discussion see [20].

Different methodologies have been created based on *i** concepts and modelling techniques. In particular the *i** framework has been exploited in different areas such as organizational modelling, business process reengineering and requirements engineering. Moreover, some proposals have been made that incorporate *i** modelling concepts to deal with software systems requirements representation and design. An example of these proposals is Tropos [8], an agent-oriented software development methodology. The contribution of Tropos at the requirements stage and in agent-oriented design has been acknowledged by different comparative studies [21-23]. Also relevant is URN [24], an *i** variation which has been added as part of the industrial Telecommunications Standard Z.151 [25] for systems specification. Besides these three main proposals, namely seminal *i**, Tropos and GRL [26], there are also others that have introduced several constructs in the language with different research aims, such as security and trust concerns [12, 18], temporal operators [27] and traceability constructs [28], among others.

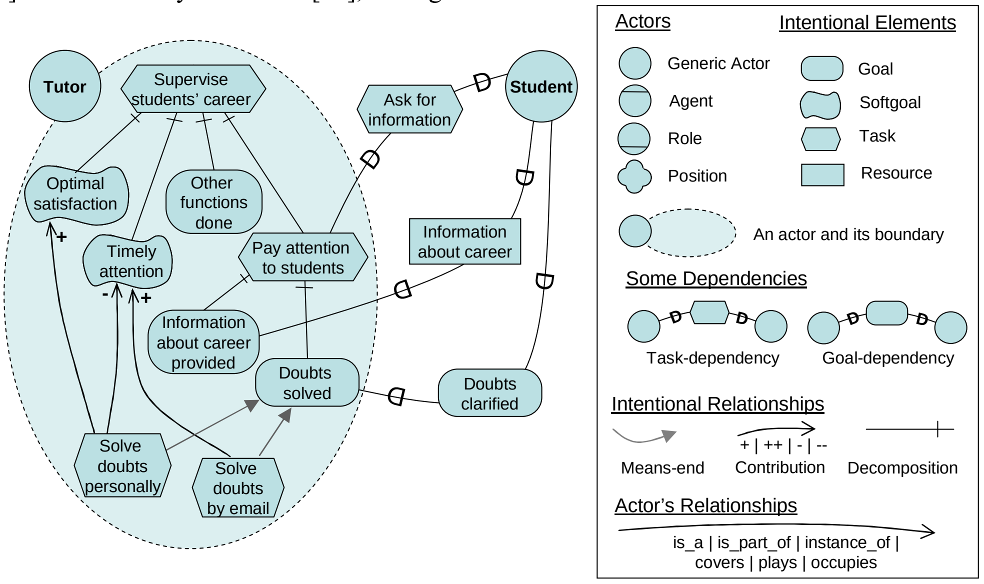


Fig. 1. Excerpt of an *i** model for an academic tutoring system.

In spite of the different aims of the proposals using the *i** framework, it is possible to classify the *i** extensions or modifications with respect to the constructs they customize (see Table 1). A more in-depth discussion may be found at [9]. We have identified only one proposal [28, 29] which generates a different language structure because it adds softgoals to describe dependency's security properties. However, even this proposal is built upon the same conceptual framework.

Table 1. Variations of the *i** framework

Variation (additions or modifications)	*i**-related proposals
Intentional elements(s)	[30-33]
Relationship(s) between actors	[8]
Relationship(s) between intentional elements	[26, 30, 33-35]
Attribute(s) for existing relationships	[36]
Attribute(s) for dependencies	[26, 36]
Constraints over the *i** framework	[8, 26, 28, 30, 31, 34]
Others	[37]

Many of these proposals have inspired the development of *i**-based software tools. In [9] there is a summary of *i**-based tools which shows a big diversity in their objectives and language details as well as limited interoperability capabilities.

In general, existing *i**-based tools and development frameworks are not capable to interoperate, i.e. interchange models and diagrams, which prevents taking advantage of existing functionalities. One of the main reasons related to the lack of interoperability of different *i**-based frameworks is that few of them have exporting capabilities to formats which allow importing the result in another tool. Moreover the different *i**-based proposals, as we show in Table 1, add or modify the syntax or even the semantics of the seminal *i** language constructs which means an additional barrier to interoperability.

As mentioned in the introduction, we have previously generated conceptual frameworks [15, 38] which help to conceptualize the different *i** variations. For example we have successfully communicated ST-TOOL [36], a software tool supporting Secure Tropos [39] with TAOM4E [15], a Tropos-based CASE tool. These tools can interchange information using a specific format based on the conceptual framework presented in [9]. The result of this experience cannot be generalized to other situations because in general the underlying tool metamodels will be different.

Given the limited nature of this set of variations and following the study [40] which proposes a set of *i** shared concepts, we have confronted the problem of defining a common interoperability language for our research community.

2.2 Formalizing the i interoperability problem*

Given that we have a set of implicit or explicit different metamodels supporting the different *i** variations, we can formalize the interoperability problem at the level of metamodels. We use the metamodel formalism notation presented in [41]: given a metamodel μ:

- $C(\mu)$ represents the set of concepts defined in μ (i.e., its concrete classes).
- $I(\mu)$ represents all its valid instances i.e., the models built from $C(\mu)$ that satisfy all the constraints stated in μ.
- Given a subset $C \subseteq C(\mu)$, $I_C(\mu)$ represents the set of instances of μ restricted to the concepts of C. It holds that $I_C(\mu) \subseteq I(\mu)$.

Given *k i** metamodel variations, denoted $\mu_1, \ldots, \mu_k$, the interoperability problem can be stated as: definition of the mapping functions $\varphi_{i,j}$: $I(\mu_i) \rightarrow I(\mu_j)$ between all pair of metamodels such that any instance in $I(\mu_j)$ can be translated into another instance of $I(\mu_j)$. Two questions remain open: 1) when these mapping functions exist, and 2) how many mapping functions need to be provided.

For the first point, it is clear that not any arbitrary pair of metamodels may be related by mapping functions because we are requiring $dom(\varphi_{i,j}) = I(\mu_i)$. A closer look to the metamodel variations reported in [41] shows that there are two types of relationships among them. On the one hand, semantic-preserving relationships that imply that μ_i and μ_j are variants modulo $\varphi_{i,j}$, i.e., $\varphi_{i,j}$ is really a bijection, e.g. renamings. On the other hand, increasing or decreasing transformations that require a value of $\varphi_{i,j}$ different from identity. But even in these cases, we have found that differences are so minor that this mapping may be effectively defined in virtually all cases.

For the second case, assuming that all the mapping functions $\varphi_{i,j}$: $I(\mu_i) \rightarrow I(\mu_j)$ exist, $1 \leq i \leq k$, $1 \leq j \leq k$, the simplest solution would be to define all these mappings explicitly. As a result, we obtain $k \times (k-1)$ mapping functions. Furthermore, when a new metamodel μ_{k+1} appears, $2 \times k$ mappings, $\{\varphi_{i,k+1}: I(\mu_i) \rightarrow I(\mu_{k+1})\}$ and $\{\varphi_{k+1,j}: I(\mu_{k+1}) \rightarrow I(\mu_j)\}$, need to be defined from the new metamodel to the existing ones and vice versa. An alternative approach consists on defining a reference super-metamodel μ_{SM} that mediates among the $\mu_1, \ldots, \mu_k$ existing ones. This is the iStarML solution. This way, the number of mapping functions is $2 \times k$, i.e. $\{\varphi_{i,\mu SM}: I(\mu_i) \rightarrow I(\mu_{SM})\}$ and $\{\varphi_{\mu SM,j}: I(\mu_{SM}) \rightarrow I(\mu_j)\}$, whilst any new metamodel requires just two new mappings to be defined. Of course, for this approach to work out, a fundamental property is required, namely that the result is the same:

$$\forall i, j: 1 \leq i \leq k \wedge 1 \leq j \leq k \wedge dom(\varphi_{i,j}) = I(\mu_i):$$
$$\forall x: x \in I(\mu_i): \varphi_{\mu SM,j}(\varphi_{i,\mu SM}(x)) = \varphi_{i,j}(x)$$

The property is taken into account as a fundamental one in the iStarML language requirements presented in the next section. Figure 2 summarizes the dimension of the *i** interoperability problem and the dimension of a solution assuming the existence of a common super metamodel.

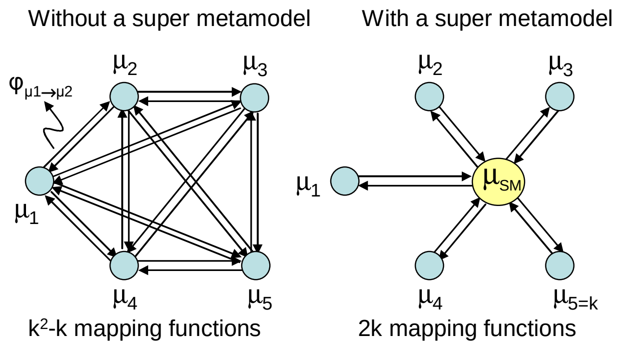


Fig. 2 The *i** interoperability problem dimension without and with a super metamodel.

3. iStarML Language Requirements

We explore below which requirements shall fulfil the iStarML language in order to be adopted by the community:

- Expressiveness. iStarML shall allow the representation, at least, of the most known versions of *i** language and, also, the design of language variations.
- Extensibility. The iStarML structure shall allow extending the language with new *i** constructs, and/or considering new aspects of existing constructs.
- Filterability. iStarML elements shall be easily separable among different criteria in order to perform adequate analysis. It means that new elements (due to extensibility) shall be described as part of knowing language constructs in order to allow their filterability.
- Flexibility. iStarML shall allow representing incomplete *i**-related information, even incomplete diagrams, and shall allow a tool to process *i** diagrams even if they include some constructs not directly treatable by that tool.
- Minimality. iStarML elements shall constitute a minimal set of constructs for representing the required knowledge on *i**.
- Simplicity. iStarML structure shall be easily readable by humans, with language elements corresponding as much as possible to the agreed names of the selected *i** constructs.
- Stability. The main iStarML elements shall represent mature and stable *i** constructs. As a result, the language shall represent the maturity of *i** established along its temporal use.

To support these requirements, we propose iStarML to be an XML-based interchange file format. Nowadays XML is the *de-facto* interchange format in Internet and it is being used in many different disciplines [42]. The XML language is based on tags which could be nested and mixed with text data. Also the tags admit attributes for keeping track of properties. Moreover, for defining specific languages using this structure, it is possible to use different Schema Languages [43]. Also there are many software tools and complementary languages (e.g. XPath [44]) which help to create, parsing and process any XML-based language. iStarML will use a set of core concepts at its heart, and then variations will be implemented in terms of these stable concepts.

The stated requirements are then fulfilled:

- Being iStarML a XML-based language contributes to the goals of *flexibility* (XML allows specifying optional structures), *filterability* (the use of some known XML query languages, such as XPath [9], allows selecting particular elements in an *i** diagram), *extensibility* (by the redefinition or use of extensible XML data types) and *expressiveness* (XML optional attributes also allow representing the current and future variations of the language).
- To use a core set of stable *i** concepts contributes to *stability* (iStarML focuses in the most mature concepts, i.e. those concepts which have been used into the different *i** related proposals with the same meaning), *minimality* (a core set means that there is not redundancy of concepts and, therefore, redundancy of language constructs) and *simplicity* (having a reduced set of clear and differentiable concepts contributes to an easy understanding of the language).
- To implement *i** variations in terms of stable concepts fixes a relevant implementation strategy that makes possible both to keep the focus on a set of mature and abstract concepts and, at the same time, to include *i** language variations as options of this core set. Thus, it contributes to *extensibility* (a broad door is kept open in order to represent language variations) and *expressiveness* (under the same schema, it is possible to represent current language variations). As a side-effect, filterability becomes possible because both variations and new elements can be filtered because the supporting language structure is known.

Finally we have explicitly considered two additional constructs for the language. On the one hand, given the highly graphical nature of *i**, we have included a construct for describing the graphical appearance of an *i** model component (e.g., position, size) so contributing additionally to *expressiveness*. On the other hand, we have also included a construct for delimitating diagrams. This diagram construct contributes to *expressiveness*, because different diagrams can be represented in the same file. Also it contributes to *simplicity*, because in order to share a detailed view of a diagram or to transfer several diagrams, only one file is be necessary.

As a result of this requirements analysis, we end up with two open questions left. First, determining the set of *i** core concepts and second, to design the precise form that the iStarML specification takes. We tackle these issues in the two next sections.

4. Determining a set of core concepts

As it was previously mentioned, in spite of the existence of different variations of *i**, there is a set of constructs which we may consider mature. Our previous work on the analysis of *i** metamodels [9, 15] have oriented us towards a core set of stable *i** abstract concepts and so to obtain a limited set of concepts which constitutes the basis of the existing *i** variations.

The core concepts have been formulated from [40] by making this metamodel more extensible: all specialization constraints were changed from complete to incomplete allowing easier addition of new subclasses; non-universal integrity constraints (e.g., restrictions on types of intentional elements) were removed; and Links were abstracted from InternalElements to IntentionalElements allowing thus the definition of links between dependums (and thus, dependencies). The resulting metamodel is presented in Figure 3.

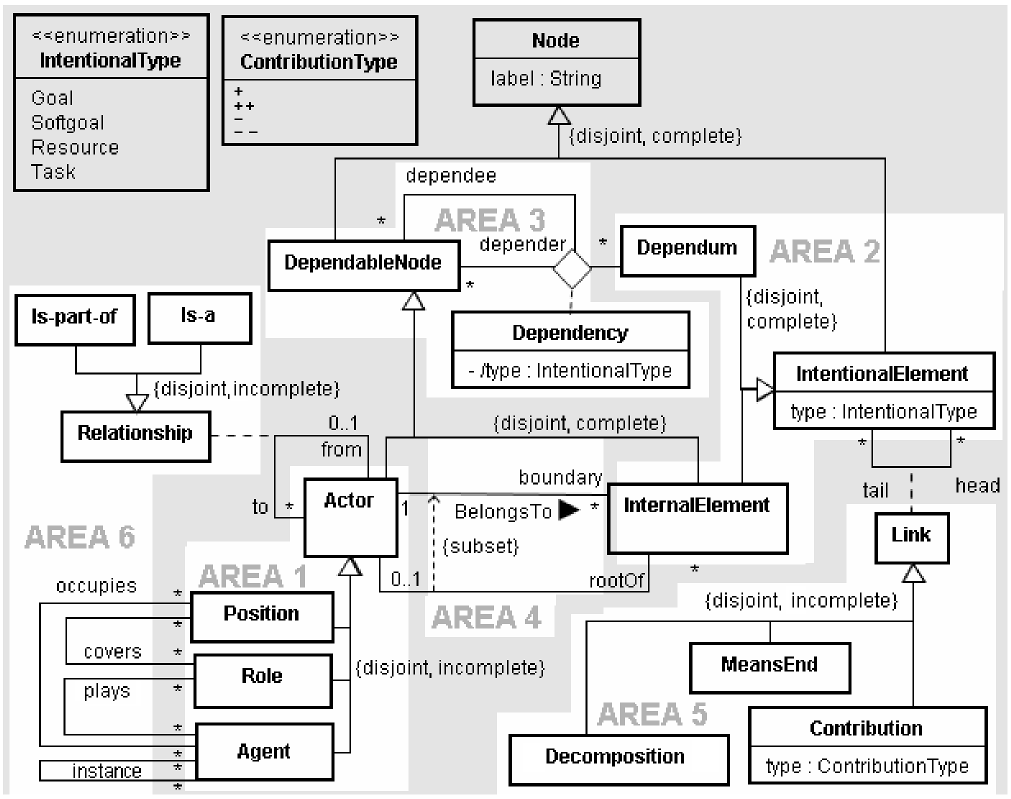


Fig. 3. The core concepts in the context of the *i** metamodel

We may distinguish up to six different parts that are highlighted in the figure and that yield to six types of core concepts: (a) actor (area 1), for representing organizational units, humans or software agents; (b) intentional element (area 2), for representing the set of elements which give rationality to the actor's actions, e.g. goals and tasks; (c) dependency (area 3), for representing actors' dependencies in order to accomplish their own goals; (d) boundary (area 4), for representing the scope of actors; (e) intentional element link (area 5), for representing the relationships among intentional elements such as means-end or decomposition relationships; and (f) actor association link (area 6), for representing the relationships among actors such as is_part_of and is_a, among others. We have considered each area as a category of core concepts that drive the structure of iStarML. Table 2 summarizes this result. The first column has the core concept name and identifies the corresponding labeled area in the metamodel (Fig 2). The second column describes the core concept. The next two columns are related to the iStarML specification which is explained in the next section.

Table 2. iStarML abstract core concepts, tags and variation representations

Abstract core concept	Core Representation	Tag	Variation Representations
Actor (Area 1)	An *actor* represents an entity which may be an organization, a unit of an organization, a single human or an autonomous piece of software.	<actor>	By using the *type* attribute, traditional actors' specializations (role, position or agent) or new actors' types can be specified.
Intentional element (Area 2)	An *intentional element* is an entity which allows relating different actors that conform a social network or, also, expressing the internal rationality elements of an actor.	<ielement>	By using the *type* attribute, traditional (goal, softgoal, resource and task) or other intentional elements can be configured. The attribute *state* can be used to specify an open set of intentional satisfactibility values.
Dependency (Area 3)	A *dependency* is a relationship which represents the explicit dependency of an actor (depender) respect to the other actor (dependee).	<dependency> <dependee> <depender>	By using the *value* attribute on tags *dependee* and *depender* an open set of dependency features can be configured.
Boundary (Area 4)	A *boundary* represents a group of intentional elements. The common type of boundary is the actor's boundary which represents the vision of an omnipresent objective observer with respect to the actor's scope.	<boundary>	By using the *type* attribute, other explicit viewpoints (different from an omnipresent observer) can be added. No *i** variation has this feature but we think that including subjectivity is a natural extension to intentional models. This attribute could handle some extension like that.
Intentional element link (Area 5)	An *intentional element link* represents an *n*-ary relationship among intentional elements (either in the actor's boundary or outside).	<ielementLink>	By using the *type* and *value* attributes, traditional (decomposition, means-end and contribution) and new relationships can be represented. For example an or decomposition can be represented setting *type* to "decomposition" and *value* to "or"
Actor association link (Area 6)	An *actor relationship* is a relationship between two actors.	<actorLink>	By using the *type* attribute, traditional (is_a, is_part_of, plays, occupies, covers and instance) and new and less used actors' relationships can be represented

5. The iStarML Specification

Once core concepts and their options have been identified, we have confronted the task of generating the iStarML specification. To do that, we have associated each core concept to an XML that represents it. Additionally the variations of each core concept are represented using attribute values. Table 2, two columns on the right, shows the result.

In addition, we have included the two explicit constructs initially considered for representing *i** diagrams and graphic expression. This action was attained by defining the corresponding tags <diagram> and <graphic>. In the first case, iStarML design allows many *i** diagrams being represented in the same file. In the second case, the <graphic> tag is a nested structure which specifies the graphic features that allow a graphic display of the *i** elements. In order to support complex graphic expressions we have extended the graphic specification to include SVG expressions, which is a XML-based graphic language gaining popularity [45]. The detailed syntax of iStarML has been described in the iStarML Reference's Guide [8].

In order to illustrate the proposal we present a simple example of using iStarML representing a small Tropos diagram [46]. In Figure 4 we show a goal dependency (G) which involves actor A as depender and actor B as dependee.

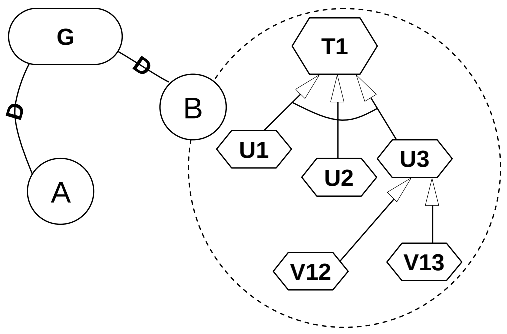


Fig. 4. Tropos's diagram for the iStarML example

The interpretation of this diagram is that the intention of the actor A to accomplish the goal G depends of the actor B. The actor B has a boundary that includes a set of tasks. There are two decompositions: an and-decomposition (U1, U2 and U3) and an or-decomposition (V12 and V13). In Tropos the line crossing the arrows indicates this difference. We assume that in this case we are not interested in the graphical representation of the model, i.e. which coordinates do they occupy in the view, which size do the elements have, etc.

For the diagram in Figure 4 we have developed the corresponding iStarML code which appears in Figure 5. In this example we show the use of the general iStarML and diagram tags. We show the use of the boundary tag in the case of actor B. Both the type of intentional link and the type of intentional element are specified by adding attributes to the core concepts corresponding to the ielementLink and ielement tags.

Also we show the corresponding decompositions using nested XML structures. Note that a change on the value attribute in the ielementLink tag could extend the decomposition type to new types of decompositions or, even, a change on the type attribute could extend the set of relationships to new conceptualizations. Finally the goal-dependency is specified as a nested structure including both dependee and depender.

```
<istarml version="1.0">
  <diagram name="Tropos's decomposition example">
    <actor id="100" name="A">
    </actor>
    <actor id="200" name="B">
      <boundary>
        <ielement name="T1" type="task">
          <ielementLink type="decomposition" value="and">
            <ielement name="U1" type="task"/>
            <ielement name="U2" type="task"/>
            <ielement name="U3" type="task">
              <ielementLink type="decomposition" value="or">
                <ielement name="V12" type="task"/>
                <ielement name="V13" type="task"/>
              </ielementLink>
            </ielement>
          </ielementLink>
        </ielement>
      </boundary>
    </actor>
    <ielement type="goal" name="G">
      <dependency>
        <depender aref="100"/>
        <dependee aref="200"/>
      </dependency>
    </ielement>
  </diagram>
</istarml>
```

Fig. 5. The iStarML code of the above diagram

6. Using iStarML in Practice

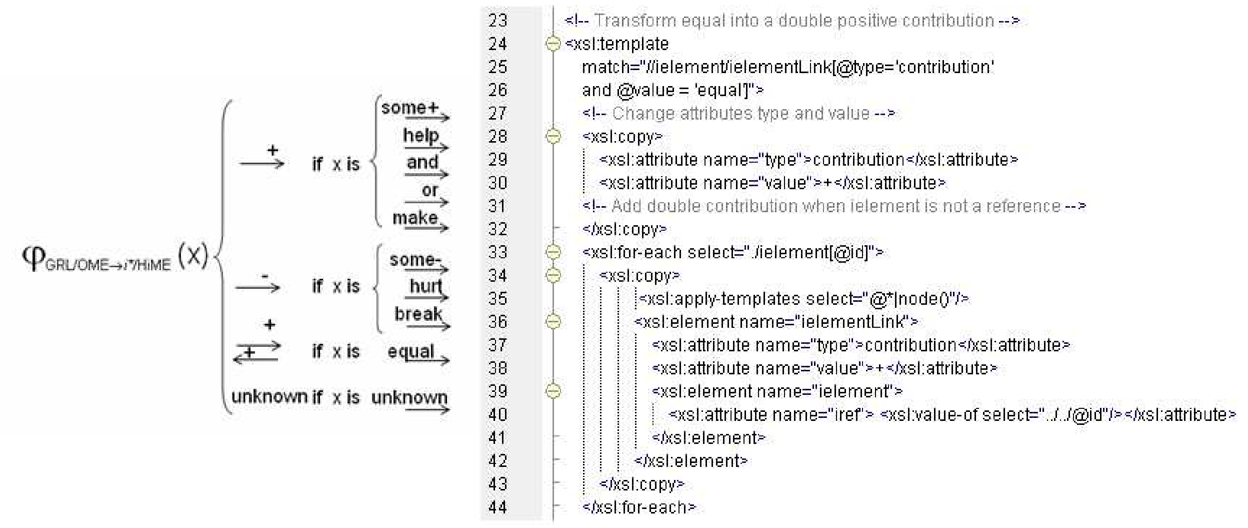
In this section we present some scenarios that may benefit from the use of iStarML and a more detailed example that shows the concrete mapping between two metamodels both at the theoretical and practical levels.

Interconnection of two modelling tools for the same dialect of *i**. An example is the interconnection of the REDEPEND tool (developed by City University) [47] and the H*i*ME tool (developed by Technical University of Catalonia) [48] both of them compliant to the seminal *i** definition by Eric Yu. The main purpose here is to share models developed in each site. In general, one may expect just minor problems concerning the core concepts that may be solved with minor effort.

Interconnection of two modelling tools for different dialects of *i**. An example is the interconnection of the OME tool (http://www.cs.toronto.edu/km/ome/) used for editing GRL models (developed by the University of Toronto) and H*i*ME. In this case, the core concepts may have more important discrepancies. For instance, GRL has 10 types of softgoal contributions (HURT, BREAK, etc.) whilst *i** in H*i*ME has just 3 (+, –, unknown). Therefore, when translating from GRL in OME to *i** in H*i*ME some accuracy is lost (e.g., both MAKE and HELP are translated into +, although MAKE is stronger than HELP). More difficult is the translation from *i** to GRL. In order to avoid false statements, a + contribution in *i** has to be translated into the weakest positive contribution in GRL, i.e. SOME+, which means “the contribution is positive, but the extent of the contribution is unknown”, which in fact reflects the meaning that + has in *i** models.

In order to enable this scenario we have developed a transformer from OME (telos) format file to iStarML (http://www.lsi.upc.edu/~ccares/ometoistarml/ccIstarmlTransformation.html).

Under this scenario we can also take advantage from the XML technology, for example if we want to translate contributions from GRL to *i** under a specific mapping function, we could use XSLT technology to translate the iStarML representation of the GRL diagram into a iStarML representation of a *i** diagram. In Figure 6 we show an example of mapping function from GRL/OME contributions to *i**/HiME contributions at the left-hand side and, at the right-hand side, the corresponding XSLT transformation to change an iStarML file from a GRL version to an *i** version.



```
<!-- Transform equal into a double positive contribution -->
<xsl:template
  match="//ielement/ielementLink[@type='contribution'
  and @value = 'equal']">
  <!-- Change attributes type and value -->
  <xsl:copy>
    <xsl:attribute name="type">contribution</xsl:attribute>
    <xsl:attribute name="value">+</xsl:attribute>
  <!-- Add double contribution when ielement is not a reference -->
  </xsl:copy>
  <xsl:for-each select="./ielement[@id]">
    <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
        <xsl:element name="ielementLink">
          <xsl:attribute name="type">contribution</xsl:attribute>
          <xsl:attribute name="value">+</xsl:attribute>
          <xsl:element name="ielement">
            <xsl:attribute name="iref"> <xsl:value-of select="../../@id"/></xsl:attribute>
          </xsl:element>
        </xsl:element>
    </xsl:copy>
  </xsl:for-each>
```

Fig. 6. Mapping function from GRL to H*i*ME version of *i**, excerpts: formal definition (left) and XSLT file using iStarML (right)

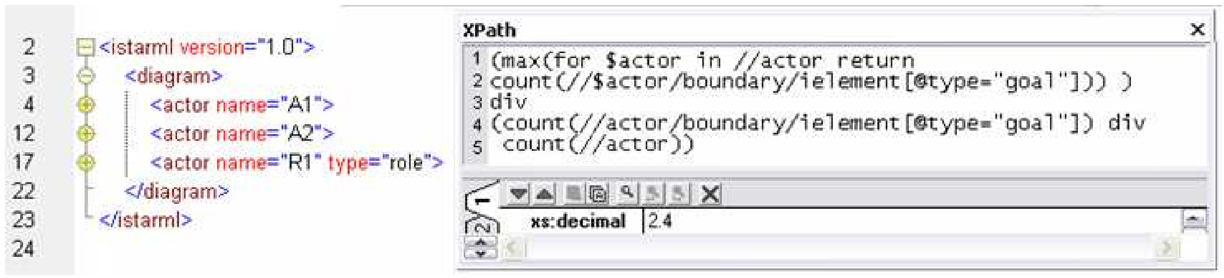
Interconnection of two *i-related tools that have different purposes**. A simple example would come again considering REDEPEND and the J-PR*i*M tool [47] not just as modelling tools but also their own capabilities. For instance, among the facilities of REDEPEND we find the ability of generating a system requirements document in textual form from an *i** model. On the other hand, J-PR*i*M aims at supporting system reengineering: starting from a use-case-based description of the system, an *i** model is generated and then alternatives may be explored and compared using metrics. Interconnecting both tools will allow generating the requirements document of the reengineered system. If we work following the other direction, J-PR*i*M facilities for computing metrics may be applied over models built with REDEPEND.

Interconnection of an *i-related tool with a different kind of tool**. This situation arises when we want to implement a goal- or agent-oriented view over some other paradigm. As a kind of example, there are several recent approaches that propose to extract variation points in feature models from goal-oriented models according to some properties [49, 50]. We have developed an implementation of this case by connecting H*i*ME with the Decision King tool [51]. Variation points are defined from elements like softgoals, means-end decompositions and is-a relationships.

Interconnection of an *i-related tool with an XML tool**. It is a particular case of the former scenario. In a few words, the *i** community may take advantage of the great deal of existing tools available in the XML community.

For the sake of brevity, we just illustrate the last scenario with two short examples. The first example is about goal and actor metrics [44, 51]. For instance, a good indicator could be the relation between the load of the most goal-loaded actor and the ideal situation of balanced goal load. If we have an iStarML representation of the analysis domain then we could use XPath [52], in order to calculate the metric value. In Figure 7 we show a reduced view of a specific iStarML file (goals are hidden) that could have been obtained from any modelling tool, and the XPath sentence which allows obtaining the pretended value. In the case we can say that the most goal-loaded actor is 2.4 more goal-heavy than a balanced situation.

In this case we observe the simplicity of the resulting query and its direct relation with explicit concepts of the *i** framework.



```
<istarml version="1.0">
  <diagram>
    <actor name="A1">
    <actor name="A2">
    <actor name="R1" type="role">
  </diagram>
</istarml>
```

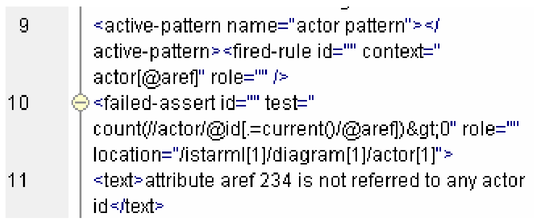
```
(max(for $actor in //actor return
count(//$actor/boundary/ielement[@type="goal"])) )
div
(count(//actor/boundary/ielement[@type="goal"]) div
 count(//actor))
```

```
xs:decimal  2.4
```

Fig. 7. Goal and actor metric calculation using XPath

The second example is about parsing iStarML files. As we suggest above, we have implemented a Schematron [53] schema. This is a rule-based syntax checker which allows customizing error messages. Applying a XSL transformation the iStarML Schematron specification produces a XSL file (istarml.xsl) which allows verifying iStarML files.

In Figure 8 we illustrate the Schematron output for the parsing of an iStarML file. In the output report we show a case of an activated rule belonging to a specific pattern (fired-rule tag on line 9). On line 10, there is a failed rule however, which adds an error message when the rule is not accomplished.



```
9   <active-pattern name="actor pattern"></
    active-pattern><fired-rule id="" context="
    actor[@aref]" role="" />
10  <failed-assert id="" test="
    count(//actor/@id[.=current()/@aref])&gt;0" role=""
    location="/istarml[1]/diagram[1]/actor[1]">
11  <text>attribute aref 234 is not referred to any actor
    id</text>
```

Fig 8. A partial output of an Schematron iStarML parser

These interoperability examples show how iStarML, XML and *i** tools can interoperate in order to reach typical goals into the software process.

7. The *i** community's perception on iStarML

In order to measure a first community perception we applied a survey about current uses of the *i** framework.[1] The population was defined as the *i** research and development community. For this reason we applied the survey on a session of the Third International *i** Workshop[2] which constituted the sample. The workshop had around 30 participants but there were more than 50 authors belonging to approximately 15 different working groups, and therefore the resulting sample size was 15. We assumed a homogenous population, this means that traditional social variables such gender or age were not considered significant on interoperability opinions or iStarML adoption attitudes. Therefore we have not stratified the population and, under this assumption, the sample of researchers is representative. This sample may seem small but in fact it represents the core of the community of *i** researchers and developers. The community of *i** users is much wider but the type of knowledge they have about *i** interoperability-related issues will be in general not so detailed as to provide highly confident answers to the questions.

In the survey, we asked for the specific perception of the interoperability problem, general knowledge about iStarML and its possibilities for overcoming the interoperability problem and, finally, we asked for the general willingness of adopting iStarML on their current research or practical applications.

The instrument is mainly based on 5-degree Likert scales, from "strongly agree" to "strongly disagree". However, the questions about iStarML adoption were formulated describing different explicit adoption attitudes.

For the data processing we followed the statistical recommendations given in [54], i.e. we selected a multinomial approach for answer evaluation of the Likert scales. It means that, for each question, we obtained five proportions, the proportion of those that answered "strongly agree" (35%), the proportion of those that answered "agree" (15%), etc. However, this way we did not arrive to any significant conclusion because confidence intervals resulted very wide and too much overlapped ([10%, 60%]). Then we applied a binomial approach, a particular case of multinomial but considering only two proportions. This means converting the 5-degree answers into success-fail answers as recommended by [53], which means losing the grade of agreement or disagreement, but it allows getting narrowest confidence intervals.

We considered three cases of data interpretation: agree answers were considered successful ones; disagree answers were considered fail answers; and, given the number of answers checking "neither agree nor disagree" ("nor"-answers), we considered the half of them as successful answers and the other half as fail answers. This option means choosing the maximum variance for a binomial case (0,25=0,5*0,5). Under these considerations we used a first test by applying the simple and rough interval of Fitzpatrick and Scott recommended in [55], and then the interval of Agresti-Coull, recommended in [56] to get confidence intervals for binomial proportions. Moreover, given that the Agresti-Coull confidence intervals allow small sample sizes (starting from 12), we added a third analysis, this time discarding the "nor-answers". In Figure 9 we show the resulting Agresti-Coull's confidence intervals using a probability of 95% (alpha = 0.05).

These results confirm our initial hypothesis because they point out that there is a shared vision about the existence of an interoperability problem. Even the worst case indicates that more than the 60% of the population recognizes the problem. Moreover, at least (i.e., considering again the worst case) the 52% of the population agrees that iStarML overcomes the problem. If we consider the center of the interval then the different population proportions appear very relevant.

[1] The instrument is available at http://www.lsi.upc.edu/~ccares/surveyistarml1.php
[2] http://www.cin.ufpe.br/~istar08/site/

	Proportion that agrees that there is an interoperability problem	Proportion that agrees that interoperability *i** mechanisms would help them	Proportion that agrees to use some *i** interoperability mechanisms	Proportion that agrees that iStarML tackles the interoperability problem	Proportion that agrees that including iStarML brings them benefits	Proportion of declared first adopters
	Agresti-Coull 95% “nor-answers” included					
upper limit	0,975	0,993	0,917	0,980	0,984	0,908
lower limit	0,609	0,644	0,508	0,526	0,555	0,429
interval width	0,367	0,348	0,409	0,454	0,429	0,479
	Agresti-Coull 95% “nor-answers” discarded					
upper limit	1,007	1,039	1,048	1,002	1,048	1,055
lower limit	0,646	0,718	0,628	0,543	0,628	0,511
interval width	0,362	0,321	0,420	0,459	0,420	0,544

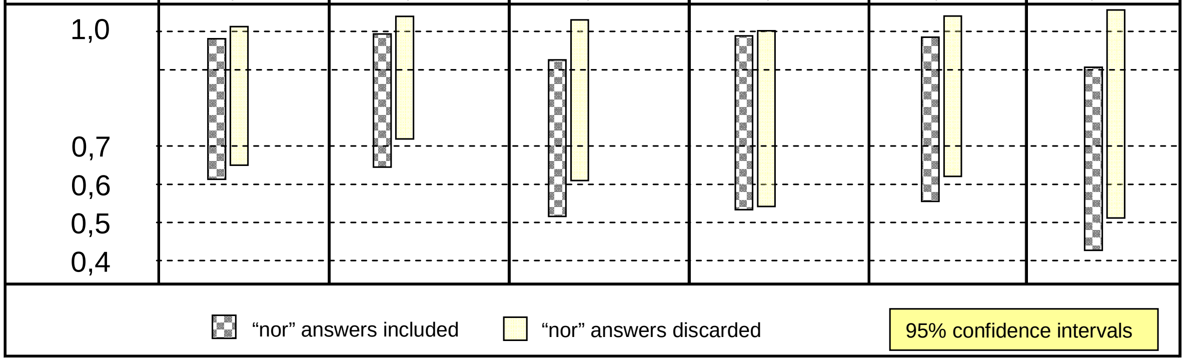


Fig 9. Confidence intervals of the *i** community perception about interoperability and iStarML.

About the answers related to adoption (either any generic interoperability mechanism or specifically the iStarML proposal) also yield good results. In this case, we have a worst case representing, at least, a 42.9% of declared first adopters. If we analyze this proportion, under the Rogers’s innovation adoption theory [47], we can see that first adopters normally became 13.5% and the “early majority”, after first adoptions, became 34% of the population. Therefore, although this 42.9% of declared adopters does not seems to be very high with respect to the whole *i** community, on the light of Rogers’s theory it seems one of the most optimistic findings of this survey.

If we analyze the possible deviation of the results due to the considered population, we remark again that workshop attendees constitute a sample formed by “special people”: at least innovators and experts. Therefore, the adoption tendency showed by the survey would correspond to leaders’ attitudes (as an opposition to followers) and their opinions would correspond to experts’ judgments Consequently, even under this hypothetical scenario, , i.e. assuming that the sample is not representative, then we can affirm that we are in presence of expert leaders, then, we can say that the qualitative and validated technique of experts’ judge can be used, therefore we do not find a different conclusions under a different (qualitative or quantitative) scenario. Moreover the conclusions coming from a qualitative perspective give additional support to these findings

8. Conclusions

In this paper, we have provided a broad justification of the iStarML proposal as a conceptual vehicle enabling interoperability inside the *i** community, both from a model-oriented perspective, and also from a tool-oriented perspective. The first one aims to provide a shared interpretation of *i** core constructs and variations and, the second one aims to develop and share *i** tools enabling a common way of storing and transmitting *i** models. The practical impact of iStarML in the *i** community may be summarized as: (i) support to model interchange among different tools; (ii) composition of existing tools to create new, complex functionalities; (iii) extending existing tools with new *i**-based analysis components; (iv) developing *i**-based analysis algorithms independently of dialect issues; (v) representing specific additional syntactic constraints to specify evolutions or new variations; (vi) having a common way of representing the differences and similarities between existing *i** variations

Interoperability can be reached even by the different *i** variants because the iStarML proposal is abstractly formulated on the core *i** concepts. This makes possible not only sharing models between different variants but also with new variants which can be built using this set of abstract core concepts. Moreover, we have carried out an empirical approach to assess the *i** community perception about the interoperability problem and the iStarML proposal. The findings confirm our initial perception about the existence of an interoperability problem and, moreover, there is an initial positive evaluation about iStarML and its capability of tackling the referred interoperability problem. Besides, the results reveal an optimistic scenario for future adoptions. At this respect we have presented two local adoptions that have been developed on our research-related tools [17, 57] and we have outlined other possible cases.

In terms of related work, as far as we know, iStarML is the first proposal aiming in this direction and accounting with a previous approval from different scholars from RE and agent-oriented software engineering communities.

About the technological implementation it is necessary to mention that not any type of schema can be applicable for the XML implementation of iStarML. For example widespread proposals such as DTD [37] and XSD [58] are not useful for the iStarML because the specification power of these proposals is restricted to elements (tags) but they do not allow specifying attribute rules which is necessary in order to express the different *i** variations. Derived from this, the XML Metadata Interchange (XMI) [59], a proposal which allow specifying generic diagrams interchange formats, is not a choice either. That is because XMI requires a MOF metamodel [24] which does not allow a flexible specification of dependencies among attributes. The only choice allowing using XMI also implies to explicitly represent intentional elements (e.g., as our Tropos MOF model in [60]), however this means losing flexibility which only be compensated by adding tags, which would made the language more complex and applicable only to some *i** variations. On our previous experiences of using XMI on Tropos tools [61, 62], we have taken into account the benefits of its technological support, but also of its difficulty, not for generating, transmitting and reading the information, but on easily understanding its contents and creating queries or transformations on it. Therefore we consider this proposal an evolution from our first attempts using XMI.

Also in term of related work Model Driven Architectures, or MDA [59] is an approach for software development for building and transforming models along the software life cycle. These models follow a set of specifications such as UML [63], MOF [24], CWM [64] and XMI for UML Models. All of them have been generated by the Object Management Group (http://www.omg.org). As far as we know, even novel domain modelling proposals from OMG like Business Process Modelling Notation (BPMN) [65] do not consider neither goal modelling, quality cross cutting concerns nor agents' intentionality modelling. That is because goal-oriented and agent-oriented conceptual frameworks have been out of the paradigmatic modelling boundaries of object orientation. Therefore we claim that iStarML is a good first step applying model-driven principles (models' transformation and interoperability) inside the *i** goal-oriented and agent-oriented conceptual frameworks. On the boundary, i.e. on the transformation of *i** and Tropos models to object-oriented modelling, different approaches have been already proposed [66]. If we try some automatization approach of these transformation proposals, for sure that iStarML would be a relevant option as Platform Independent Language (PIM). Therefore, we claim that iStarML enables MDA principles beyond object orientation. Besides, if we consider not only an agent-oriented design but also an agent-oriented implementation, iStarML can enable MDA principles on software methodologies which consider agent or goal-orientation as methodological choice.

As the main barrier to overcome for adopting iStarML, of course semantic integration is the most important. First of all, we remark again that, as we have already reported in [9], differences in the core of *i** are minor. For those cases in which direct translation from one *i** framework to another is not direct, we need to develop point-to-point translations as mentioned in Section 6 for the case of soft goal contributions. Other constructs may be treated similarly, e.g. trust relationships in trust models may be translated to dependencies. For others, they may be just discarded, e.g., the traceability construct "supports" as defined in [33] is just syntactic sugar and may be removed without losing meaning. However, a few of existing proposals have not an obvious treatment (e.g., the temporal relationship among tasks as proposed in Formal Tropos) and therefore the only action to take is to detect and report these situations, probably keeping them as annotations in the generated models.

In terms of future work we will keep supporting iStarML adoptions because they will allow materializing interoperability benefits in the *i** community, e.g. using goal-oriented domain models on agent-oriented software design, implementing repositories of requirements patterns, using agent-oriented metric tools on actor-oriented business modelling, and a long and unpredictable list of human activities resulting of the capacity of sharing models and tools that work on a common but customizable conceptual framework. Precisely, in order to clarify these possibilities, we have already started a study about interoperability scenarios for using iStarML in the context of software development process.

References

[1] P. Zave, Classification of Research Efforts in Requirements Engineering, ACM Computing Surveys, 29 (4) (1997) 315-321.

[2] A. v. Lamsweerde, Requirements Engineering in the Year 00: A Research Perspective, Proceedings of the International Conference on Software Engineering, ICSE 2000, Limerick Ireland, Jun 4-11, (2000) 5-19.

[3] J. Mylopoulos, L. Chung, E. Yu, From Object-Oriented to Goal-Oriented Requirements Analysis, Communications of the ACM, 42 (1) (1999) 31-37.

[4] A. Dardenne, A. v. Lamsweerde, S. Fickas, Goal-directed requirements acquisition, Science of Computer Programming, 20 (1-2) (1993) 3-50.

[5] E. Yu, Modelling Strategic Relationships for Process Reengineering, Computer Science, University of Toronto, Toronto (1995).

[6] E. Yu, J. Mylopoulos, N. Maiden, P. Giorgini, *Social Modelling for Requirements Engineering*. Boston: MIT Press, To be published, 2008.

[7] X. J. Mao, E. Yu, Organizational and social concepts in agent oriented software engineering, Lecture Notes in Computer Science, 3382 (2005) 1-15.

[8] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents And Multi-Agent Systems, 8 (3) (2004) 203-236.

[9] C. Ayala, C. Cares, J. P. Carvallo, G. Grau, M. Haya, G. Salazar, X. Franch, E. Mayol, C. Quer, A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages, Proc of the Conf on Software Engineering and Knowledge Engineering (SEKE2005) (2005) 43-50.

[10] M. Kolp, P. Giorgini, J. Mylopoulos, Organizational Patterns for Early Requirements Analysis, Lecture Notes in Computer Science (CAiSE'03), 2681 (2003) 617-632.

[11] G. Gans, G. Lakemeyer, M. Jarke, T. Vits, SNet: A Modeling and Simulation Environment for Agent Networks Based on i* and ConGolog, Lecture Notes in Computer Science (CAiSE'02), 2348 (2002) 328-343.

[12] H. Mouratidis, M. Weiss, P. Giorgini, Security Patterns Meet Agent Oriented Software Engineering: A Complementary Solution for Developing Secure Information Systems, Lecture Notes in Computer Science (ER2005), 3716 (2005) 225 - 240.

[13] R. Guizzardi, G. Guizzardi, A. Perini, J. Mylopoulos, An Ontological Account of Agent-Oriented Goals, Lecture Notes in Computer Science, 4408 (2007) 148-164.

[14] H. Estrada, A. Martínez, O. Pastor, J. Mylopoulos, An Experimental Evaluation of the i* Framework in a Model-based Software Generation Environment, 18th Int. Conf. on Advanced Information Systems Engineering (CAISE 06), Luxembourg, June, (2006) 513-527.

[15] A. Susi, A. Perini, J. Mylopoulos, P. Giorgini, The Tropos Metamodel and its Use, Informatica, 29 (2005) 401-408.

[16] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Requirements Engineering Meets Trust Management, Model,

Methodology, and Reasoning, Lecture Notes in Computer Science, 2995 (2004) 176-190.

[17] D. Bertolini, A. Novikau, A. Susi, A. Perini, TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. Issue on the development process, FBK-irst, http://sra.itc.it/tools/taom4e/file/demo//TAOM4E_technical_report.pdf (accessed March 2008) (2006).

[18] L. Liu, E. Yu, J. Mylopoulos, Security and Privacy Requirements Analysis within a Social Setting, International Conference on Requirements Engineering (RE'03), Monterey, California, USA, September, (2003) 151-161.

[19] A. G. Sutcliffe, S. Minocha, Linking Business Modelling to Socio-technical System Design, Lecture Notes in Computer Science (CAiSE'99), 1626 (1999) 73-87.

[20] E. Yu, J. Mylopoulos, N. Maiden, P. Giorgini, *Social Modeling for Requirements Engineering*. Boston: MIT Press, To be published, 2010.

[21] L. Cernuzzi, M. Cossentino, F. Zambonelli, Process models for agent-based development, Engineering Applications of Artificial Intelligence, 18 (2) (2005) 205–222.

[22] K. H. Dam, M. Winikoff, Comparing agent-oriented methodologies, Lecture Notes in Computer Science, 3030 (2003) 78-93.

[23] J. Sudeikat, L. Braubach, A. Pokahr, W. Lamersdorf, Evaluation of Agent–Oriented Software Methodologies – Examination of the Gap Between Modeling and Platform, Lecture Notes in Computer Science (AOSE'04), 3382 (2005) 126-141.

[24] D. Amyot, G. Mussbacher, URN: Towards a New Standard for the Visual Description of Requirements, Lecture Notes in Computer Science, 2599 (2003) 21-37.

[25] Z.151: User Requirements Notation (URN) - Language Definition, ITU-T, http://www.itu.int/rec/T-REC-Z.151-200811-P/en (accessed Oct 2009) (2009).

[26] GRL - Goal Oriented Requirement Language, http://www.cs.toronto.edu/km/GRL/ (accessed Jan 2009) (n.d.).

[27] A. Fuxman, M. Pistore, J. Mylopoulos, P. Traverso, Model Checking Early Requirements Specifications in Tropos, Proceedings of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada (2001) 174-181.

[28] H. Mouratidis, J. Jürjens, J. Fox, Towards a Comprehensive Framework for Secure Systems Development, Lecture Notes in Computer Science (CAiSE'06), 4001 (2006) 48-62.

[29] G. Grau, A Comparative of *i** Modelling Tools, http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Tools (accessed Jan 2009) (2006).

[30] C. Cares, X. Franch, E. Mayol, Extending Tropos for a Prolog Implementation: A Case Study Using the Food Collecting Agent Problem, Lecture Notes in Computer Science (CLIMA VI), 3900 (2006) 396-405.

[31] P. Donzelli, A goal-driven and agent-based requirements engineering framework, Requirements Engineering, 9 (1) (2004) 16-39.

[32] G. Gans, M. Jarke, S. Kethers, G. Lakemeyer, Modeling the Impact of Trust and Distrust in Agent Networks, Third International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2001), Interlaken, Switzerland, June 4, (2001)

[33] X. Franch, G. Grau, E. Mayol, C. Quer, C. Ayala, C. Cares, F. Navarrete, M. Haya, P. Botella, Systematic Construction of *i** Strategic Dependency Models for Socio-technical Systems, International Journal of Software Engineering and Knowledge Engineering, 17 (1) (2007) 79-106.

[34] J. Castro, M. Kolp, J. Mylopoulos, A Requirements-Driven Development Methodology, Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001,Interlaken, Switzerland(2001) 108-123.

[35] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, Specifying and analyzing early requirements in Tropos, Requirements Engineering, 9 (2) (2004) 132-150.

[36] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling Social and Individual Trust in Requirements Engineering Methodologies, Lecture Notes in Computer Science, 3477 (2005) 161–176.

[37] D. C. Fallside, P. Walmsley, XML Schema Part 0: Primer, http://www.w3.org/TR/xmlschema-0/ (accessed Jan 2009) (2004).

[38] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, ST-Tool: A CASE Tool for Security Requirements Engineering, Proc. of the 13th IEEE Conference on Requirements Engineering (RE'05), Paris, France, Aug 29 - Sept 2, (2005) 451-452.

[39] D. Bertolini, A. Siena, TAOM4E and the Tropos Reasoners. Issues on the integration process, ITC-Irst, Trento, Technical Report 2006.

[40] C. Cares, X. Franch, E. Mayol, C. Quer, A Reference Model for i*, in *Social Modeling for Requirements Engineering,*, E. Yu, J. Mylopoulos, N. Maiden, and P. Giorgini, Eds.: MIT Press, 2010

[41] G. Wachsmuth, Metamodel Adaptation and Model Co-adaptation, Lecture Notes in Computer Science, 4609 (2007) 600-624.

[42] L. Dongwon, W. C. Wesley, Comparative analysis of six XML schema languages, SIGMOD Rec., 29 (3) (2000) 76-87.

[43] Altova, XMLSpy user manual and programmers' reference., Altova Gmbh, Technical Report 2005.

[44] J. Clark, S. DeRose, XML Path Language (XPath) Version 1.0, World Wide Web Consortium (W3C), http://www.w3.org/TR/xpath (accessed Jan 2009) (1999).

[45] C. Cares, X. Franch, A. Perini, A. Susi, iStarML Reference's Guide, Technical University of Catalonia, Report LSI-07-46-R, http://www.lsi.upc.edu/~techreps/files/R07-46.zip (accessed Jan 2009) (2007).

[46] G. Grau, X. Franch, N. Maiden, REDEPEND-REACT: an Architecture Analysis Tool, Proc. of the 13th IEEE Conference on Requierements Engineering, Paris, France, Aug 29 - Sept 2, (2005) 455-456.

[47] G. Grau, X. Franch, S. Ávila, J-PRiM: A Java Tool for a Process Reengineering i* Methodology, 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis, MN, USA, Sept. 11-15, (2006) 359-360.

[48] L. López, X. Franch, J. Marco, HiME: Hierarchical i* Modeling Editor, Proc. of the 28th Int. Conf. on Conceptual Modeling (Tool Demo Session), Gramado, Brazil, Nov 9-12, (2009)

[49] B. González-Baixauli, M. A. Laguna, J. C. S. P. Leite, Using Goal Models to Analyze Variability, First Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS 07), Limerick, Ireland, January 16–18, (2007) 101-107.

[50] D. Dhungana, P. Grünbacher, R. Rabiser, DecisionKing: A Flexible and Extensible Tool for Integrated Variability Modeling, Proc of the First International Workshop on Variability Modelling of Software-intensive Systems, Limerick, Ireland, January 16-18, (2007) 119-126.

[51] X. Franch, On the Quantitative Analysis of Agent-Oriented Models, Proc. of the 18th Conf. on Advanced Information Systems Engineering (CAiSE'06), Luxembourg, Jun 5-6, (2006) 495-509.

[52] R. Jelliffe, The Schematron Assertion Language 1.5, Academia Sinica Computing Centre, http://xml.ascc.net/resource/schematron/Schematron2000.html (accessed September 2007) (2002).

[53] R. Göb, C. McCollin, M. F. Ramalhoto, Ordinal Methodology in the Analysis of Likert Scales, Quality & Quantity, 41 (2007) 601-626.

[54] R. Peck, L. D. Haugh, A. Goodman, *Statistical Case Studies: A Collaboration Between Academe and Industry*: SIAM, 1998.

[55] L. D. Brown, T. T. Cai, A. DasGupta, Interval Estimation for a Binomial Proportion, Statistical Science, 16 (2) (2001) 101-133.

[56] E. M. Rogers, *Diffusion of Innovations*, 4th ed. New York: Free Press, 1995.

[57] J. Bosak, T. Bray, D. Connolly, E. Maler, G. Nicol, C. M. Sperberg-McQueen, L. Wood, J. Clark, W3C XML Specification DTD ("XMLspec"), http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm (accessed Jan 2008) (1998).

[58] MOF 2.0/XMI Mapping Specification, v2.1, Object Management Group - OMG, http://www.omg.org/docs/formal/05-09-01.pdf (accessed Jan 2009) (2005).

[59] Unified Modeling Language Specification, Object Management Group - OMG, http://www.omg.org/docs/formal/05-04-01.pdf (accessed Jan 2009) (2005).

[60] D. Bertolini, L. Delpero, J. Mylopoulos, A. Novikau, A. Orler, L. Penserini, A. Perini, A. Susi, B. Tomasi, A Tropos Model-Driven Development Environment, Forum Proceedings CAiSE, Luxembourg, June 5-9, (2006)

[61] L. Penserini, A. Perini, A. Susi, J. Mylopoulos, Agent Capability: Automating the Design to Code Process, Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS'06), Lisbon, Portugal, December 14-15, (2006)

[62] A. Brown, An introduction to Model Driven Architecture, IBM, http://www.ibm.com/developerworks/rational/library/3100.html (accessed March 2008) (2004).

[63] Meta Object Facility (MOF) 2.0 Core Specification, Object Management Group - OMG, http://www.omg.org/docs/ptc/04-10-15.pdf (accessed Jan 2009) (2003).

[64] G. L. Zuñiga, Ontology: Its Transformation From Philosophy to Information Systems, Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA (2001) 187 - 197.

[65] Business Process Modeling Notation Specification, Object Management Group (OMG), http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf (accessed Jan 2009) (2006).

[66] F. Alencar, B. Marín, G. Giachetti, O. Pastor, J. Castro, J. H. Pimentel, From i* Requirements Models to Conceptual Models of a Model Driven Development Process, Proc. of the 2nd IFIP WG8.1 Working Conference on The Practice of Enterprise Modeling, PoEM 2009, Lecture Notes in Business Information Processing, vol39, Stockholm, Sweden, November 18-19, (2009)