Making Explicit some Implicit *i** Language Decisions

Lidia López, Xavier Franch, Jordi Marco

Universitat Politècnica de Catalunya (UPC)
c/Jordi Girona, 1-3, E-08034 Barcelona, Spain
{llopez, jmarco}@lsi.upc.edu, franch@essi.upc.edu

Abstract. The *i** (i-star) framework is one of the most widely adopted modelling approaches by several communities (business modelling, requirements engineering, ...). Probably due to its highly strategic nature, the definition of the modelling language offered by the framework does not make explicit the full behaviour of some basic constructs, leaving them thus open to several interpretations. This looseness may not be important in some contexts, even it may be beneficial since it leaves room for researchers to customize the framework to their needs. However, it becomes an obstacle in other situations, e.g., model interoperability and model-driven development. In this paper we identify ambiguities and silences in the *i** language definition in a systematic manner, and then we propose an interpretation to deal with them. In some cases, the proposal may include the addition of some annotation into some language construct. The result is a formal definition taking the form of a UML conceptual data diagram (a metamodel) with several important integrity constraints.

Keywords: *i** framework, i-star, iStar, ambiguity, silence.

1 Introduction

The *i** (pronounced *eye-star*) framework [1] is currently one of the most widespread goal- and agent-oriented modelling and reasoning frameworks. It has been applied for modelling organizations, business processes and system requirements, among others.

Throughout the years, different research groups have proposed variations to the modelling language proposed in the *i** framework (for the sake of brevity, we will name it "the *i** language"). Some variations come from paradigm shifts (e.g., using *i** for modelling services [2], see [3] for a compilation), others propose some particular type of new construct (e.g., for dealing with security aspects [4]), but others just issue slight modifications related to the core of the *i** language (from now on, "the *i** language core"). This third type of variations mainly appear because the definition of the *i** language core is loose at some parts, and researchers may have interpreted the same constructs in different ways. The absence of a universally agreed metamodel has accentuated this effect [5].

Looseness is partly due to the high strategic nature of the *i** framework: the emphasis is more on high-level concepts like actors, goals and dependencies, than on low-level details. Thus, it may be argued that the mentioned ambiguities and silences may not be important in some contexts (e.g., interactive creativity meetings with stakeholders). Being this true, it also happens that there are other contexts in which ambiguities and silences become an obstacle:

- Model interoperability. Accepting a community scenario in which different variants for *i** exist, supporting the interchange of models and the interconnection of tools seems to be a reasonable goal. In a previous work [6], we have reported how different interpretations of the *i** language core may hamper and eventually prevent automatic model interchange among tools.
- Model-driven development (MDD). Several works are addressing the use of *i** diagrams as starting point in MDD processes [7]. Due to their very nature, MDD processes require models with a clear and non-ambiguous meaning.
- Precise definition of some *i** constructs. In the available definitions of the *i** language core, some constructs were not defined up to the last level of detail. For instance, in [8] we have explored the *i** subtyping construct at the level of Strategic Rationale (SR) diagrams. In this work, we have identified some looseness that forced us to make some decisions about the *i** language.

Our position is that each and every modeling language definition should be complete and consistent regardless of its intended use. This is the ultimate motivation of our work. Those contexts that require a more informal or agile use should lead to lighter versions of the language but still these should be complete and consistent, as well as compatible with the full version.

The work presented here addresses these problems and specifically tries to answer the following research questions:

- RQ1. Which ambiguities and silences exist in the current definition of the *i** language core?
  - RQ1.1. What constructs can be considered to form the *i** language core?
- RQ2. What decisions can be made to solve these ambiguities and silences?
  - RQ2.1. Is it necessary to include additional features in the *i** language core to implement these decisions?
  - RQ2.2. Are there particular issues that deserve further research before an informed decision can be made?
- RQ3. What is the final form that an ambiguity- and silence-free *i** language core definition should take?

The rest of the paper is structured as follows. Section 2 enumerates and analyses the sources consulted. Section 3 shows the methodology used to drive our work. Sections 4 to 7 are the core of the work, complemented with section 8 where we formulate our final proposal for the *i** language core. Section 9 provides the conclusions and future work. Basic knowledge of *i** is assumed, see [1] and the *i** wiki [9] for details.

2 Background: Analysis of the *i** Framework

There is a great deal of research made by the *i** community that is relevant to our objectives. This section tries to summarize the most important observations after analysing eight types of sources. It is clear that we cannot aim at giving details in the paper for every individual type of source, so we have decided just to enumerate the most significant sources considered in each type and provide in the second subsection a global consolidation of observations.

2.1 Sources of our work

The types of inputs considered have been the following:

Main dialects. Arguably, we may consider that there exist three main streams of *i** variants: 1) the seminal proposal of the *i** framework [1], updated in [9]; 2) the Goal-oriented Requirement Language (GRL) which is part of the User Requirements Notation (URN) [10]; 3) Tropos, an agent-oriented software methodology that adopts *i** as its modelling language [11].

Metamodels. Several contributions exist that propose metamodels for *i** with different purposes: 1) metamodels for particular *i** variants like GRL's [12] and Tropos' [13]; 2) metamodels for supporting model interoperability [14]; 3) metamodels for providing a reference framework [15][16]; 4) metamodels as the basis for tool construction [17].

Literature. In [14] we performed a literature review as a baseline for model interoperability analysis. Since we focused on the analysis of *i** constructs, the results of the review are applicable also in this paper. The review was conducted over the following conferences and journals for the period 2006-2010: ER, CAiSE, REJ, DKE, IS Journal, RE, RiGiM, WER, *i** workshop, and it included also the recent book on *i** [3]. After some filtering, we selected 63 contributions proposing addition, removal or modification of basic *i** constructs as described in Table 1.

Tools. We experimented with the some of the most (if not the most) used *i** modeling tools: Open OME [18], jUCMNav [19], REDEPEND [20] and TAOM4E [21]. Being modelling tools, they necessarily provide (intentionally or not) answers to some of these ambiguities and silences.

Techniques. Similarly to tools, what makes interesting the definition of techniques is that they provide an interpretation to all *i** constructs. Two main types of techniques exist, evaluation procedures [22][23] and qualitative reasoning techniques [24].

Evaluation reports. Some works exist that similarly to our aim, have provided an analysis of different aspects of *i**: 1) Analysis of current uses, best practices and misunderstandings [25][26]; 2) direct comparison of several major proposals [15][16]; 3) reflections about *i** subordinated to the analysis of its visual notation [27]; 4) definition of a model interchange format [28].

Real experiences. The experiences of *i** in real projects provides insights about how the framework has been used in industrial projects, cf. [29][30][31] among others.

Personal feedback. Last but not least, interaction with researchers in the community, discussions, attendance to talks, etc., had provided us useful insights on the use of *i**.

Table 1. Variations proposed by the *i** community in the last 5 years (selected venues only). Each paper increments each column at most in 1.

	Actors	Actor links	Dependencies	Intentional elements (IE)	IE links
New	4	24	10	21	21
Removed	8	5	2	1	0
Changed	3	1	1	36	43

2.2 Observations

We have built our work from the analysis of the sources above. The first output is the decision of which elements form the *i** language core. The rationale adopted is: an element is included in the *i** language core if it is adopted or accepted (sometimes implicitly) by all the sources mentioned above. To start with, we consolidate this core into a preliminary UML conceptual data schema that includes classes for the core concepts and associations among them (see Fig. 1). In the next sections we will add the required information into this conceptual schema until we reach a non-ambiguous and complete definition of the core.

The data schema shows the three key concepts of *i**: `Actor`, `Dependency` and `IntentionalElement`. Actors have a `boundary` that includes their `InternalElements`, a subtype of intentional element. Both actors and internal elements may be related through `Links`. Concerning dependencies, they connect `DependencyParticipant` (either actors or internal elements) acting as `dependers` or `dependees` through some `Dependum` (another type of intentional element). We remark that in this initial model we are not including any type of constraint (even we do not include cardinality), since for all the concepts we may find several interpretations that could violate these constraints, regardless how general we try to be.

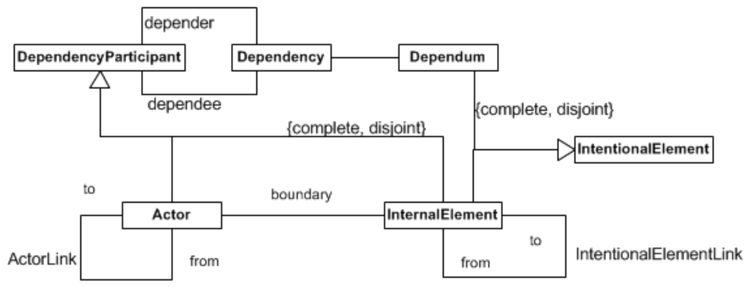


Fig. 1. The *i** language core: preliminary representation as a UML conceptual data schema.

3 Systematic Management of Ambiguities and Silences

To proceed further in the formulation of the core of the *i** language, we wanted to apply some systematic process to ensure the completeness of the analysis. We opted for designing a set of questions to apply to each element in the core. Since elements are represented in a data schema, we derived these questions from the analysis of the UML superstructure [32] regarding to the metaelements that we will use in the core definition: classes, associations, specializations and attributes.

Table 2 summarizes the most important questions. In the analysis, we left out from the very beginning information that appears in the UML metamodel but that it is not of interest in our context, e.g. questions related to visibility. We have not shown in the table either some questions that were applied but that have provided a consistent result in all cases (e.g., we first had two questions to find out if some association or attribute were derived, or also to check for overlapping specializations, but since the answer was always negative, we do not show them). Concerning the rest of questions:

Table 2. Summary of questions for ambiguity and silence systematic detection.

Concept	Id	Question
Class	C1	Is the class specialized according to some criteria (type, ...)?
	C2	For each criterion in C1, is the specialization complete or incomplete?
	C3	Does the class have any attribute?
Associations	A1	Which are the characteristics of the different roles (cardinality, ordering, ...)?
	A2	Is the association of a specific type (composition, aggregation)?
	A3	Is it an association class? (if so, Class' questions also apply)
Attributes	P1	Which type is it?
	P2	Which is the cardinality (univalued, multivalued, optional, ...)?
All types	T1	Is there any other constraint involving one or more elements of the model?

– For classes, most questions are related to their possible participation in specialization hierarchies: possible classification criteria and their completeness. The last question refers to the existence of attributes.
– Concerning associations, the questions are designed to find out: characteristics of roles; nature of the association (e.g., aggregation, composition...); and if it is an association class (and then class questions also apply).
– For attributes, fundamental questions are about their type and cardinality.
– Last, a general question was applicable to all type of elements, searching for properties that relate different elements of the model or express characteristics of one particular type of element (e.g., reflexivity, transitivity, ...).

The next sections discuss the application of these questions to the different parts of the data schema shown in Fig. 1, although we will not show continuously the questions and how they were applied. References to elements of the final metamodel presented in Section 8 are written in Courier style.

4 Actors and Actor Links

Question C1 applied over `Actor` could be matter of discussion, since not all the proposals found in the literature propose distinguishing types of actors. However, those that do, adhere to the classical distinction: `Role`, `Position`, `Agent`. The most remarkable exception is GRL/URN in all its sources (language report, metamodel and jUCMNav tool) but even here we may find some ambiguity, since at some documents it is said that the type of actor can be stated using metadata, therefore we conclude that introducing `ActorType`s does not cause a severe conflict with GRL/URN.

Question C2 over actors reveals one typical situation of silence in most sources. However, we find some examples in which non-classified actors coexist with actors of a particular type. Another argument that could be given is that if we consider the model development process, even if we want to end up with a model with all actors belonging to a given type, intermediate states could still keep actors whose type is still not determined. Therefore, we allow `General` actors to coexist with specialized ones.

When it comes to `ActorLink`, first of all we find several `ActorLinkType`s in the literature. Those sources that distinguish actor types, also provide the classical links `plays`, `occupies` and `covers`, applied to the correct types (`IC5-7`). In addition, specialization (`is-a`) and aggregation (`is-part-of`) are often mentioned and for those sources that don't, it seems more that simply it was not an objective of the work than an intended decision. Therefore, we also propose them. The main problem at this moment is the instance (`INS`) relationship among agents. This concept appears in the seminal Yu's proposal but it is not formally defined, it is just used in the examples. A fundamental question arises here: is it really a construct that must appear at the model itself, or does it belong to a different modelling level? In some sense, referring to the MOF specification [33], it could be argued that agents' instances belong to M0 whilst the rest of an *i** model is at M1. Therefore, we do not include the instance relationship in our current core proposal, instead we formulate our open issue (OI):

OI1: Does the `instance` relationship belong to the *i** core?

When we explore the `ActorLink` association (A1 and A2, and also T1), we find the typical scenario that we feel justifies the need of this paper. There are very permissive scenarios that do not include any constraint (e.g., the OpenOME tool) and others that state some concrete rules (e.g., the Tropos metamodel). Several issues get different responses in different sources, or they are not mentioned at all:

– Cardinalities: e.g., may an actor be specialization of more than one actor (see Fig. 2, left)? May different actor links have different cardinalities?
– Constraints among types of connected actors: e.g., may an actor of a given type be a subtype or part of an actor of another type (see Fig. 2, center)?
– Simultaneous application of actor links: e.g., may an actor be at the same part a subtype and a part of two other actors, or even of the same actor (see Fig. 2, right)?

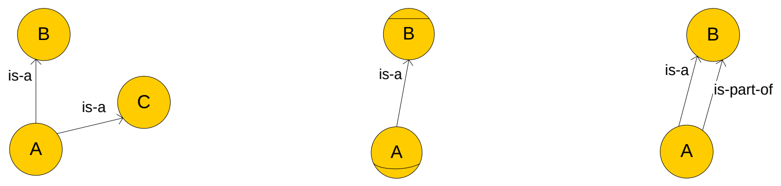


Fig. 2. Some forbidden situations in the *i** language core for actor links.

As a result of this analysis, we make the following decisions:

– We do not allow an actor being a subtype of more than one actor (`IC8`). Reason is simplicity: multiple inheritance is known as a source of confusion for modellers. Specially because, as we will see in Section 5, there are a lot of questions about the meaning of these links when intentional elements are considered.
– We do not allow neither `is-a` nor `is-part-of` to involve actors of different type (`IC3`) or to be applied to the same pair of actors (`IC4`). Not just simplicity is behind this decision, also conceptual clarity (from an ontological point of view) and compliance to community behaviour: some proposals state this constraint explicitly (remarkably the Wiki does) and those that don't, seem more not having paid attention to the issue that having made a conscious decision.
– We do not allow cycles, not just for the same type of link, but for all (`IC2`). We have not found any meaningful situation where cycles should be allowed.
– There is no special property for the `ActorLink` association. We just paid attention to the possibility of `plays` being an association derived from `covers` and `occupies`, but clearly this is not the case, since an `agent` may `play` a `role` without an intermediate `position`. Also we discarded `is-part-of` to be an aggregation.

5 Internal Elements and Internal Element Links

Internal elements configure the rationale of actors. A review of the sources shows a clear consensus in four types (albeit some minor terminological differences): `goal`, `softgoal`, `task` and `resource`. The existence of a fifth type, `belief`, is not so clear. The main dialects use it inconsistently: whilst Yu's thesis named them, in fact it does not make use of them, but then the wiki clearly defines them. URN/GRL includes beliefs from the very beginning, whilst Tropos just do it sometimes. Similar dissimilarities may be found in the rest of sources. In our core proposal, we are including beliefs in the `IEType` intentional elements' type because we think that they are really modelling something that cannot be modelled otherwise ("a condition about the world that the actor holds to be true" [9]). Following GRL metamodel, they cannot be part of a dependency (`IC9,14`). Since this could be a controversial point we identify an OI:

OI2: Does the `belief` relationship belong to the *i** core?

When it comes to internal element links (`IELinks`), we find fundamental questions that are typically answered differently by different sources:

Types of links. There is agreement on `task decompositions` (TD), `means-end links` (ME) and `contributions to softgoals` (CSG). Other variants are rare and it does not seem advisable to include them. Still some problems arise (see Fig. 3):

– For TD, the elements that decompose are considered in AND relationship. Therefore, it is not possible to decompose any other type of internal element other than task with an AND decomposition.

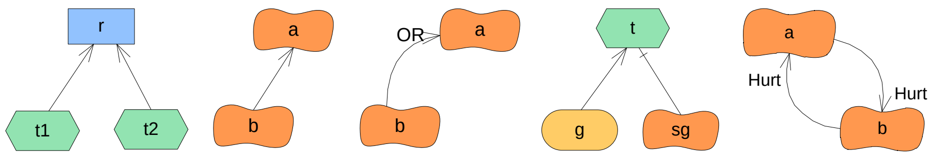
– For ME, there is no consensus about which relationships are valid between sources and targets of the links, e.g., may a resource be an end? Also, it is not clear whether the means are exclusive (XOR) or not (OR).

– For CSG, there is agreement that contributions must be typed, but there is no consensus about the admitted types. One of the most used values' domain includes also AND and OR types, and then it is not clear whether this prevents the use of softgoals as root for TD and ME.

Joint use. In general, it is not stated whether an internal element may be decomposed using more than one type of link. E.g., if it is a task, may it be both decomposed with TD and ME?

Cycles. Although in general cycles are not allowed, we have the specific case of CSG: if softgoal A contributes negatively to softgoal B and the other way round, is it possible to state both relationships together in the model?

Roots. Usually it is not stated if more than one root is possible inside an actor's boundary, although by observation of examples this seems to be the usual case. Constraints on the type of the root(s) are usually not given either.



(a) Means-end (b) Contributions to softgoals (c) Mixing links (d) Cycles

Fig. 3. Some extreme situations in *i** decomposition links.

Below we show the decisions made at this respect. We have tried to combine the following criteria: conceptual clarity, keeping the language as understandable as it could be; soundness, avoiding situations that could drive to meaningless models; flexibility, avoiding unnecessary restrictions; expressive power, trying to exploit full capabilities of goal models; alignment with community, to foster acceptance.

Types of links:

– For TD, we allow the possibility of having other types than `task` in the root, therefore the name should change to just `decomposition`. This could be a highly controversial point, but in fact note that currently: 1) decomposition of goals into subgoals is currently proposed in some documents as: first, provide a `task` means to the `goal` end, and then decompose the `task` into `goals`, which we believe just introduces some noise in the model with the same objective; 2) AND-`decomposition` of softgoals is supported by `contribution` as defined e.g. in Wiki.
– For ME, we take the most permissive option and interpret OR instead of XOR.
– In both cases, aligning with URN/GRL position, we tend to remove as many constraints as possible concerning valid types of internal elements. In other words, we want the core to be as inclusive as possible since the sources' analysis has demonstrated great diversity. Therefore, we just control the use of beliefs: a belief can be decomposed just into beliefs; a belief can contribute to softgoals (`IC10`).
– For `contribution`, we adhere to the proposal in the Wiki and URN/GRL (excluding AND/OR contributions, that we consider as decomposition, see above), a `ContType` formed by values `make`, `help`, `some+`, `unknown`, `some-`, `hurt`, `break`.

Joint use. An internal element cannot be decomposed using more than one decomposition type (`IC13`). A decomposed softgoal may be the target of contribution links.

Cycles. Only cycles among softgoals are allowed (`IC12`).

Roots. More than one root of any type is allowed.

The connexion between internal elements and actors is implemented through the `boundary` association (see Fig. 1). It is declared as composition since an internal element belongs to exactly one actor and its life is dependant of its actor's life.

A last point that needs attention is the effect of actor links on internal elements. In other words, if there is a link from actor A to actor B, what effect has this on the internal elements in A? This question arose when we used intensively the `is-a` relationship in our models and motivated a line of research that is still ongoing [8]. A lot of open issues had been to be tackled, e.g., which modifications are allowed inside A of the elements inherited from B? We foresee that the same questions may arise when considering the other types of actor links. The answer to these questions motivates an OI that in fact is many-fold (one for each type of actor link):

OI3: Which are the consequences on internal elements of actors that are related to other actors via actor links?

6 Dependencies

`Dependums` are at the heart of dependencies and their nature is one of the most agreed concepts in *i**: they have the same type `IEType` than internal elements but not including beliefs (`IC9`), so we adhere to this position.

When it comes to `dependencies`, we find several issues that deserve discussion:

Source and target elements. We may find all possible situations of connection among internal elements and actors depending on the internal knowledge about involved actors. Therefore, we allow any possible situation. The initial metamodel at Fig. 1 supports this decision with the `IntentionalElement` class. However, a twilight zone appears when considering what happens e.g. with a dependency among two actors when a third actor inherits from one of the former [8]. We identify an OI:

OI4: Which are the relationships between dependencies and actor links?

Strength. Basically the discussion is whether strengths should be part of the core or not. A lot of researchers in the community simply do not use strengths. However, we still believe that they have a potential and their use could be of interest in several contexts (since they provide both information about criticality and effort).

Multiplicity. A situation that is not completely specified in *i** proposals is whether a dependency can have multiple dependees or dependers and in this case, what is the meaning. We find two different style repetitions: the dependum is repeated in different dependencies; or the dependum has several links stemming or going to. After examining the existing proposals, the decision made is (see Fig. 4):

– We allow both: 1) one dependency with multiple dependers and dependees (see Fig. 4, left); 2) several dependencies with the same dependum, provided that the same pair (depender, dependee) does not appear in more than one of them (`IC14`).
– If a dependency has several dependees, then the satisfaction of the dependum depends on all of them altogether. This is the most usual intent in the models in which we have found this situation. As drawback, this decision prevents to express that a single dependee could make a dependum satisfied.
– If a dependency has several dependers, then all the dependers depend on that dependum the same way. If needed for clarity of the drawing, we admit to split the dependency into several since no ambiguity is possible (see Fig. 4, right, bottom).
– If there are two dependencies with the same dependum *d* (see Fig. 4, right, top), it means that *d* is describing some kind of entity that appears in 2 different contexts.

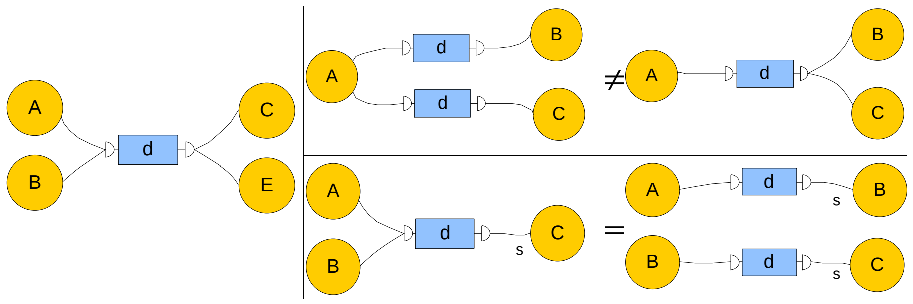


Fig. 4. General framework for dependencies in the *i** language core.

We remark that we explicitly avoid in our core some scarce uses that may be found in some proposals (more in examples than in formal definitions themselves), e.g.: dependencies involving intentional elements inside an actor, or dependency links that include some contribution value.

7 Some Addition of Information into *i** Models

In the previous analysis we have strictly adhered to consensus and intentionally avoid any proposal that could be considered out of the current trends in *i** (with the only exception of `task-decomposition` converted into `decomposition`). However, we still find some situations whose resolution cannot be implemented without adding some information in *i** models. We are identifying next these situations and providing a way of modelling them (shown in Fig. 5). Remarkably, we will provide default actions that makes the classical *i** models compliant to our definition. We also try to provide a uniform graphical view in terms of notation, as the figure shows.

Generalization sets. In some models we have needed to specialize an actor according to different criteria. For instance, in a Travel Agency case, we needed to specialize the TravelAgency actor with respect to the target type of customer (Individual, Organization) and the type of agency (Internet, Traditional). We faced two problems: first, *i** does not support the grouping of subactors by criterion, so that the model didn't catch the intended meaning; second, we were not able to distinguish the disjoint nature of the first criterion from the overlapping criterion of the second one. We propose to make *i** more expressive at this respect, adding: 1) the ability to group subactors; 2) the ability to classify the specialization according to completeness and disjointness. We propose to use the UML metamodel concept of `GeneralizationSet` to specify these issues. As default case, we choose one single `complete` and `disjoint` specialization criterion without explicit `name`.

Decomposition links. As commented in Section 4, we are proposing an AND-decomposition link beyond task decomposition, and we also have the means-end links that are a kind of OR-decomposition. Putting both things together, we propose to follow Tropos' proposal in which we have just the `Decomposition` link that can be qualified as AND or OR, and in fact we propose also XOR as a third type of qualification to support expressiveness to obtain then an attribute of type `LogicalType`. By default, AND-decomposition links are interpreted as `decomposition`, and OR- and XOR-decomposition links as `means-ends`.

Dependency links. We find a similar situation than above for dependency links in the dependee's side. We propose the same solution: to qualify the type of combination of dependees with AND, OR, XOR (attribute in `Dependency` of type `LogicalType`).

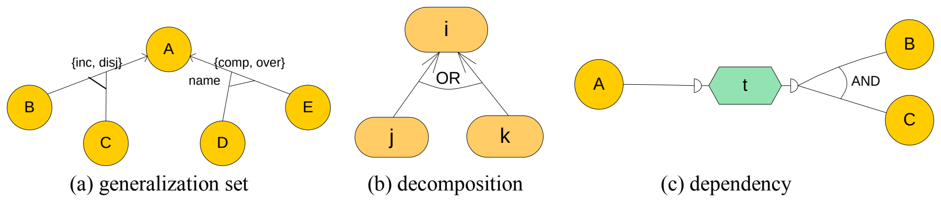


Fig. 5. Proposals for the *i** language core.

To conclude with, we identify a final OI:

OI5: How much acceptable those proposals could be in the *i** community.

8 The *i** Language Core: Final Representation

In this section we show the result of the analysis undertaken in the four previous sections. The metamodel is shown in Fig. 6 and the integrity constraints in Table 3.

It is important to remark that we have a degree of freedom when one class has some specialization criterion with respect to the representation in UML of each value of this criterion. Given a class C in which a classification criterion may take values k_1, ..., k_n, we may create k subclasses of C, one for each k_i, or we may create an Enumeration type with values k_1, ..., k_n and then an attribute of this type in C. We have decided to avoid creation of specialization relationships to keep the model simple, and thus the only situation in which sub-classes are created is when some attribute has to be defined in at least one of the k_i. This is a change with respect to some former metamodels we have proposed (e.g., [16]), but after some experience with them we think that it is worth keeping the class diagram simple. The rest of the metamodel just reflects the decisions made in the paper.

9 Conclusions

In this paper we have formulated a precise definition of the core constructs of the *i** language. We have organized the research into several research questions (see Section 1) which we hope have been satisfactorily answered:

– We have undertaken a comprehensive analysis of the existing body of knowledge for the *i** framework with focus on the language. The analysis has relied upon several types of sources. As a result, we have identified the *i** language constructs whose formal definition is not completely defined (research question RQ1) and determined which part of the language can be considered to be the core (RQ1.1).

Table 3. Integrity constraints over the *i** core language.

Id.	Concept	Integrity Constraint
IC1	name attribute	There cannot be two model elements with the same name except for:
IC1.1	InternalElement	internal elements: name restriction applies inside actors' boundary
IC2	ActorLink	Cycles are not allowed regardless of the type of ActorLink
IC3		The links is-a and is-part-of must connect actors of the same type
IC4		The is-a and is-part-of links cannot be applied to the same pair of Actors
IC5		The link occupies must connect an Agent with a Position
IC6		The link covers must connect an Position with a Role
IC7		The link plays must connect an Agent with a Role
IC8		An Actor cannot be a subtype of more than one Actor
IC9	Dependum	Dependums cannot have Belief as type
IC10	InternalElement Link	Beliefs can be decomposed only into beliefs
IC11		Contributions can only have softgoals as to
IC12		The only cycles allowed are those that involve only contribution links
IC13		An internal element can be decomposed using one type of decomposition
IC14	Dependency	A dependum cannot appear twice among the same pair of Actors
IC15		Depender and dependee actors must be different
IC16		Beliefs cannot be neither depender nor dependee

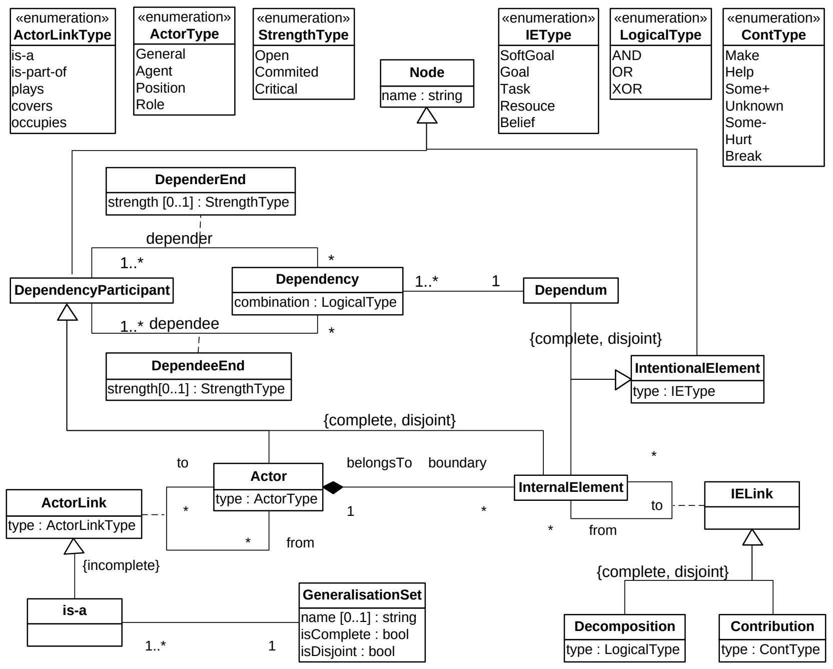


Fig. 6. The *i** language core: final representation as a UML conceptual data schema.

– We have proposed how to deal with the identified ambiguities and silences. For most of them, we have provided in this paper our interpretation of the problem and then we have made a particular decision to solve it (RQ2). In some cases, we have added and formally defined some minor notational elements to the *i** language to fulfil this goal (RQ2.1). These new elements are always optional in nature. Last, we have identified some open issues that cannot be decided in the paper but require a more detailed analysis (RQ2.2). We remark the intended decision of keeping separated RQ2 from RQ1: whilst RQ1 is answered through analysis, RQ2 is requiring decisions to be made, which can always be a matter of discussion.
– We have articulated our proposal around a UML data schema with integrity constraints to fully define the meaning of the *i** language constructs (RQ3).

As a summary, we may say that we have provided a first consolidated step towards having a community agreement of the formal meaning of *i** constructs up to a level of detail that is not currently available. As mentioned in the introduction, we think that the results of this work can be useful in several contexts that really require this level of detail, like model-driven development, model interchange and tool interoperability. We also remark that although the paper has focused on the *i** language, we believe that the method and criteria used could eventually be applied in other domains facing a similar scenario.

In this paper, we have assumed that "a" (i.e., exactly one) *i** language core exists. An open question that requires further investigation is whether different contexts could require different language cores. For instance, does the *i** language needed for conducting creativity meetings share the same core with model-driven development-oriented *i**?

At a first sight, this work may seem to contradict somehow some of our previous work on the construction of an *i** metamodel [15][5][16][14]. In these works, we advocate for an *i** metamodel general enough to host most of the proposed variants of *i**. But in fact we argue that the two approaches are complementary. In these cited works, we design a general metamodel but the *i** language core is embedded into it. What we are proposing here is to make more accurate the expression of the core. Thus putting both lines of research together, we could include the decisions made in this paper into the metamodel proposed in these sources. Thus, any new *i** variant could configure the metamodel to its own needs (e.g., by adding some new type of intentional element, or restricting the allowed types of links decomposition) but at the same time, relying on a stable *i** core with a clearly defined semantics, eventually shared by all variants.

As future work, we first mention the consideration of the open issues identified in the paper. In addition, we have identified two lines of research:

1) building an ontological foundation for the *i** core language, using a foundational ontology like UFO [34];

2) adapting existing techniques to the proposed *i** language core, which requires reflecting about the concept of satisfactibility of intentional elements.

Acknowledgments

This work has been supported by the Spanish project TIN2010-19130-c02-01. The authors want to thank Carlos Cares for his suggestions in early versions of the work; Renata and Giancarlo Guizzardi for our fruitful discussions on the topic; and the anonymous reviewers for their valuable feedback.

References

1. Yu E.: *Modelling Strategic Relationships for Process Reengineering*. PhD.Computer Science University of Toronto, Toronto (1995).
2. Estrada H., Martínez A., Pastor O., Mylopoulos J. and Giorgini P.: Extending Organizational Modeling with Business Services Concepts: An Overview of the Proposed Architecture. ER 2010.
3. Yu E., Giorgini P., Maiden N. and Mylopoulos J. (eds.): *Social Modeling for Requirements Engineering*: The MIT Press, 2011.
4. Mouratidis H., Giorgini P. and Manson G.: Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. CAiSE 2003.
5. Franch X.: Fostering the Adoption of *i** by Practitioners: Some Challenges and Research Directions. In *Intentional Perspectives on Information Systems Engineering*. Springer, 2010.
6. Colomer D., López L., Franch X and Cares C.: Model Interchange and Tool Interoperability in the *i** Framework: An Experiment. WER 2011.
7. Alencar F., Marín B., Giachetti G., Pastor O., Castro J. and Pimentel J.: From *i** Requirements Models to Conceptual Models of a Model Driven Development Process. PoEM 2009.

8. López L., Franch X. and Marco J.: Defining Inheritance in *i** at the Level of SR Intentional Elements. iStar 2008.
9. The *i** Wiki, http://istar.rwth-aachen.de. Last accessed: July 2011.
10. ITU-T Recommendation Z.151 (11/08), *User Requirements Notation (URN) - Language Definition*, http://www.itu.int/rec/T-REC-Z.151/en, 2008.
11. Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J. and Perini A.: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3), 2004.
12. Amyot D., Horkoff J., Gross D. and Mussbacher G: A Lightweight GRL Profile for *i** Modelling. ER Workshops 2009.
13. Susi, A., Perini A. and Mylopoulos J.: The Tropos Metamodel and its Use. *Informatica*, 2005.
14. Cares C. and Franch X.: A Metamodelling Approach for *i** Model Translations. CAiSE 2011.
15. Ayala C., Cares C., Carvallo J.P., Grau G., Haya M., Salazar G., Franch X., Mayol E, and Quer C.: A Comparative Analysis of *i**-Based Agent-Oriented Modeling Languages. SEKE 2005.
16. Cares C., Franch X., Mayol E. and Quer C.: A Reference Model for *i**. Book chapter in [3], 2011.
17. Lucena M., Santos E., Silva C., Alencar F., Silva M.J. and Castro J.: Towards a unified Metamodel for *i**. RCIS 2008.
18. OpenOME Tool, http://www.cs.toronto.edu/km/openome/. Last accessed, March 2011.
19. jUCMNav Tool, http://jucmnav.softwareengineering.ca. Last accessed, March 2011.
20. Lockerbie J. and Maiden N.: REDEPEND: Extending *i** Modelling into Requirements Processes. RE 2006.
21. TAOM4E Tool, http://selab.fbk.eu/taom/. Last accessed, March 2011.
22. Horkoff J. and Yu E.: Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. ER 2010.
23. Amyot D., Ghanavati S., Horkoff J., Mussbacher G., Peyton L. and Yu E.: Evaluating Goal Models within the Goal-oriented Requirement Language. *International Journal of Intelligent Systems* 25(8), 2010.
24. Giorgini P., Mylopoulos J., Nicchiarelli E. and Sebastiani R.: Reasoning with Goal Models. ER 2002.
25. Horkoff J., Elahi G., Abdulhadi S. and Yu, E.: Reflective Analysis of the Syntax and Semantics of the *i** Framework. ER Workshops 2008.
26. Webster I., Amaral J. and Cysneiros L.M.: A Survey of Good Practices and Misuses for Modelling with *i** Framework. WER 2005.
27. Moody D., Heymans P. and Matulevicius R.: Visual Syntax does matter: Improving the Cognitive Effectiveness of the *i** Visual Notation. *Req. Engineering Journal* 15(2), 2010.
28. Cares C. Franch X., Perini A. and Susi A.: Towards Interoperability of *i** Models using iStarML. *Computer Standards & Interfaces* 33(1), 2011.
29. Carvallo J.P. and Franch X.: On the Use of *i** for Architecting Hybrid Systems: A Method and an Evaluation Report. PoEM 2009.
30. Annosi A., Pascale A., Gross D. and Yu E: Analyzing Software Process Alignment with Organizational Business Strategies using an Agent- and Goal-oriented Analysis Technique. iStar 2008.
31. Maiden N., Jones S., Ncube C and Lockerbie J.: Using *i** in Requirements Projects: Some Experiences and Lessons Learned. Book chapter in [3], 2011.
32. *OMG 2.2 Unified Modelling Language Superstructure*, 2009.
33. *MOF 2.0 Core Final Adopted Specification*, 2006.
34. Guizzardi G. and Wagner G.: Using UFO as a Foundation for General Conceptual Modeling Languages. In *Theory and Application of Ontologies*, Springer, 2010.