Preliminary version, some minor differences with final version exist

Fostering the Adoption of *i** by Practitioners: Some Challenges and Research Directions

Xavier Franch

Abstract The *i** framework is a widespread formalism in the software engineering discipline that allows expressing intentionality of system actors. From the time it was issued, in the mid-nineties, a growing research community has adopted it either in its standard form or formulating variations in order to adapt it to some particular purpose. New methods, techniques and tools have made evolve the framework in a way that it may be currently considered quite mature from the scientific perspective. However, the *i** framework has not been transferred to practitioners at the same extent yet: industrial experiences using *i** are not many and have been mainly conducted by *i** experts that are part of that very research community. Therefore, it may be argued that some steps are needed for boosting the adoption of *i** by practitioners. In this chapter, we identify some scientific challenges whose overcoming could represent a step towards this goal. For each challenge, we present the problem that is addressed, its current state of the art and some envisaged lines of research.

1 Introduction

Goal-oriented modelling is a widespread technique in the software engineering community. It is used in broad disciplines like requirements engineering [68] and organizational modelling [39], and in more specific scopes as service modelling [53] and adaptive system modelling [15]. The intentional perspective on systems engineering proposed by C. Rolland [54, 53] had a great impact in the field and largely contributed to this dissemination.

Among these several existing goal-oriented frameworks, methods and languages (e.g., KAOS [13], MAP [53]), the *i** framework [71] is currently one of the most widespread modelling and reasoning approaches. It supports the construction of models that represent an organization and its processes as an intentional network of actors and dependencies, which may be decomposed into simpler elements. Reasoning techniques allow checking properties and performing some kind of qualitative [29, 35] and quantitative [19] analysis.

The intentional nature of *i** is very clearly explained by Yu: "In *i** modelling, we focus on intentional properties and relationships rather than actual behaviour.

By not describing behaviour directly, an intentional description offers a way to characterize actors that respects the autonomy premise. Conventional system modelling which offers only static and dynamic ontologies leads to an impoverished and mechanistic view of the world. Intentional modelling provides a richer expressiveness that is appropriate for a social conception of the world" [72].

Arguably, it may be said that the *i** framework has reached a high maturity level from the scientific point of view, as a result of the intensive work undertook by many research groups, leading to an increasing body of knowledge available through scientific papers and experience reports presented in world-leading journals and conferences. A growing community has been established too, with two clearly visible meeting points: a periodical event, the *i** workshop; and a working space, the *i** wiki [36]. Basic knowledge on *i** is offered through tutorials in scientific conferences and a textbook in which the community research groups provide a comprehensive revision of the state of the art [73]. Last, it may be mentioned the recognition of an *i**-based language like URN as a telecommunication standard [37].

Taking all of this research into consideration, it could be expected that the adoption of *i** by practitioners should had grown the same. Unfortunately, it is clearly not the case. A recent survey of practice [14] does not even mention *i** among current requirements engineering adopted formalisms. Just a few, though valuable, experiences have been reported on the use of the framework and associated tools. We may mention:

- The air traffic control experiences by Maiden et al., see [41, 44] as summary.
- Experiences in Ericsson Marconi Spa about knowledge transfer and process alignment [4, 5].
- The application of *i** for articulating activities around Off-The-Shelf-based and hybrid systems in the Etapatelecom Ecuadorian company [11, 12].

These experiences were basically successful, although they highlighted several obstacles on the adoption of *i** in medium- and large-scale projects. Also, it should be remarked that those experiences have been mainly conducted by expert *i** modellers, being thus uncertain to what extent novice *i** modellers (e.g., the typical profile of a requirement elicitation facilitator) are able to conduct their processes in an effective and efficient way.

Therefore, we strongly argue that the *i** community should dedicate the necessary effort on exploring effective ways to transfer the framework to practitioners, making it more usable in industrial experiences. Efforts are twofold. On the one hand, exploration of scientific issues having to be with the framework that may enhance its usability, improving thus chances of adoption by practitioners. On the other hand, planning and executing strategic community actions not directly related to scientific findings (industry-oriented seminars and tutorials, etc.). Due to the nature of this book, we will focus on the first topic.

This chapter provides an overview of some of the most relevant scientific challenges that shall be overcome in order to attain (at least partially) this knowledge transfer goal. Because of length limitations, we will concentrate on challenges re-

lated to the framework's modelling language more than to analysis techniques defined around. For each challenge, after reviewing its context, the problem to be solved and the state of the art, research directions are proposed and justified. As a result, we expect to stimulate research in the community along these directions and thus to effectively help bridging the gap among researchers and practitioners in the use of the *i** framework.

2 Challenge 1: Agreeing on the *i** Metamodel

Context. Since it was first released in 1995, the *i** framework has been adapted to the needs of specific research groups that wanted to represent concepts specific of their application domain, like security [28], temporal precedence relationships [23] or architectural concepts [31]. Furthermore, even the original framework had several variations (GRL for standardization purposes [67], the Tropos methodology on top of *i** [7]) and experienced a natural evolution on time that has led to a slightly modified version available in the *i** wiki [36]. As a result, we may conclude that there is a plethora of variations available in the community, used by several authors with different purposes (see [9] for a summary and more detailed analysis).

Problem. This diversity, although not necessarily pernicious, hampers the progress of the *i** community. When reading a work around the *i** framework, it is necessary first to understand what concrete version of *i** is being used. If the contribution is based on the original framework, sometimes the authors declare which version are they using (lately, it is happening to be the wiki version), but sometimes there is no explicit mention, which usually makes the reader a bit hesitant about details of the proposal being presented. Also it has to be said that the wiki version is currently described as an informal tutorial (a users' guide) without providing such a metamodel. On the other hand, if the work is proposing some new variation, enrichment or customization of *i**, the semantics is sometimes given informally or by using a formalism which is not easy to align with the available descriptions of *i**. Therefore, as some authors explicitly claim [8, 46], a unifying metamodel seems a must.

Challenge. The *i** framework shall include one and only one metamodel; well-established customization strategies for designing variants of this metamodel shall be used.

State of the art. We may find in the literature several approaches of *i** metamodels. Ayala et al. proposed a metamodel [6] that evolved into a more elaborated one by Cares et al. [9], see Fig. 1. It was designed by considering the features of the original Yu's version [71] (which included its own metamodel written in Telos [45]) and its two most widespread variations, GRL [67] and Tropos [7], proposing a unifying model. It is intended to be reusable too, and several superclasses appear

to support this objective. Variations are proposed to be modelled by refactoring although no exploration on the semantic consequences of this process is included. Other approaches with different aim are that of Susi et al. [62] and Roy et al. [57], presenting metamodels not general-purpose as the previous one, but tailored to the specific capabilities of the Tropos associated *i** variation and GRL, respectively.

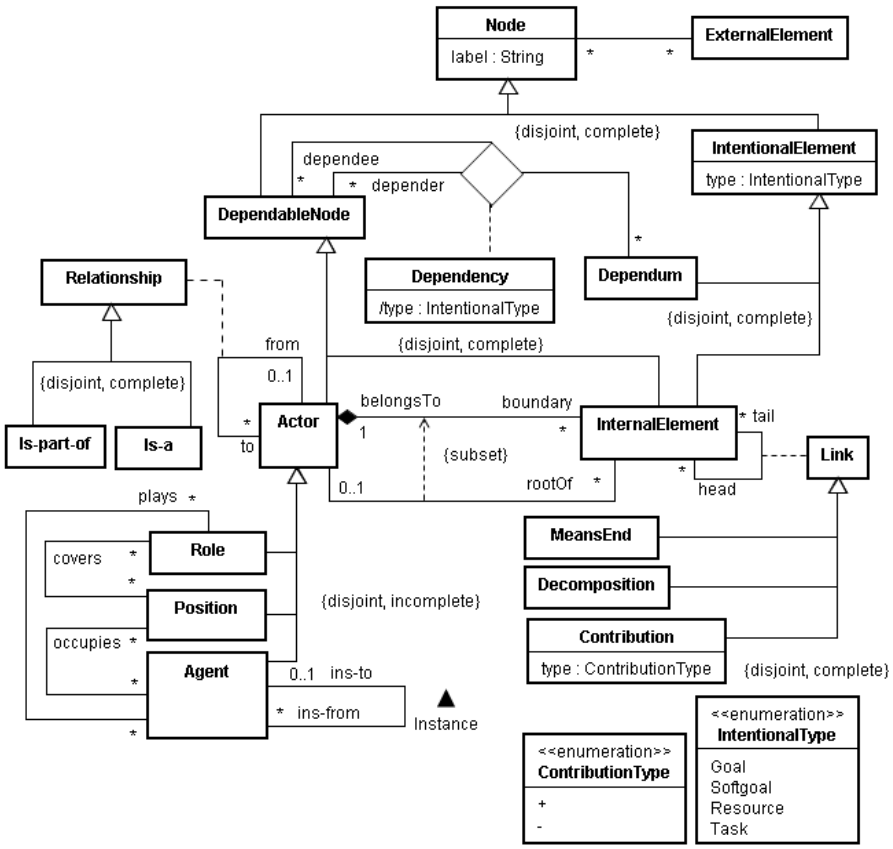

Fig. 1. The *i** metamodel as proposed in [9] (integrity constraints not included)

Research directions. Concerning the metamodel, the community should agree on the final form of the *i** metamodel (probably by upgrading into metamodel the current wiki informal description), and consider it as part of the framework core. Let's call this metamodel "the *i** metamodel". The *i** metamodel should be recognized as standard by all the community, providing then a common vocabulary for the community. We think that the most recent Cares' et al. proposal [9] is an adequate starting point since it is able to express virtually all of the concepts included in the wiki version. Also, it is not oriented to any particular metamodeling technology like EMOF, we think that technology-independence is a good property for this general-purpose metamodel.

As for the customization, we propose that each proposal which needs a particular variant of the *i** metamodel as reference, should formally specify the relationship with the *i** metamodel. To do so, more than a simple refactoring exercise as proposed in [9], a semantically rich definition has to be provided. We may take for instance the formal framework proposed by Wachsmuth [69] and then use the several relationships defined therein (equivalence, enrichment, extension, etc.) to classify the proposed metamodel with respect to the *i** metamodel, providing also the concrete mapping functions that express the differences in a rigorous manner.

One possible immediate application of this notion is to obtain technology-oriented versions of the *i** metamodel. For instance, in [26] it is proposed to use *i** as initial model in a model-driven development process, and a particular version of the metamodel using EMOF defines the form that this departing model may have. In the context proposed here, this EMOF version could be elaborated as a semantics-preserving variation of the *i** metamodel. The mapping from this EMOF-based version to the *i** metamodel would then be completely accurate.

Another technology-oriented version is the iStarML interchange format [10]. This format translates the metamodel proposed in [9] into XML. It is currently used as import/export format in the H*i*ME tool [34] and planned for adoption in a next release of TAOM4E [63]. A natural consequence of agreeing on the *i** metamodel would be to evolve iStarML into a version compliant to this metamodel.

3 Challenge 2: Providing Methodologies for *i** Modelling

Context. In general, modelling is an activity that requires prescriptive methods in order to be repeatable, reproducible and in general, predictable. One could reasonably expect that when facing the same problem, two different (teams of) experts in both the domain being modelled and the modelling framework, working independently, should produce very similar, in some sense "equivalent" models.

Problem. Modelling becomes harder when the level of abstraction of the knowledge to be modelled increases. Therefore, creating a model for the requirements of a system is harder than creating a model of a software architecture. When considering the *i** framework, the modelling activity is harder than ever due to its intentional nature. As Yu states, "The *i** framework is aimed at modelling strategic relationships and reasoning. Such knowledge is not expected to be complete" [71]. In addition, the *i** language itself allows a degree of freedom that creates some uncertainty not only to the novice, but also to the expert, modeller. A summary of recurrent questions include:

- Being *i** models of strategic nature, which elements have strategic relevance enough as to be included in the models, and which don't, and why.
- As a particular case of this situation, which are the conditions that indicate that a model is completely refined.

- In some particular situation, what *i** modelling element is the most adequate. A typical example is: when a task produces a resource, which element must be included: just the task, just the resource, none, or both.

Given an answer as accurate as possible to these and similar questions is the only way to overcome the modelling uncertainty problem.

Challenge. The *i** community shall have available a range of well-defined modelling methods as predictable as possible.

State of the art. The construction of goal-oriented models in general has been subject of interest by the requirements engineering community, for instance Rolland et al. used explored the use of scenarios to drive the discovery of goals [56]. The need of modelling methods for the *i** framework became obvious soon and a major response was the formulation of the Tropos method [7]. Tropos spans four phases of software development, namely early requirements, late requirements, architectural design and detailed design, previous to implementation (in some papers considered a fifth phase) using some agent-oriented infrastructure. The method is not prescriptive inside these four phases and then the modeller still has a great freedom in completing the model. In [27], the authors propose the use of social patterns as a way to elicit the general structure of models. However, still the patterns just provide a general layout of the models.

Other authors propose more detailed methods. In [22] the R*i*SD methodology is proposed for the modelling of Strategic Dependency (SD) *i** diagrams. It is organized into several steps, and intents to be a highly prescriptive procedure. As part of the steps, we may mention: (a) The formulation of two alternative decision trees (under dependum's nature and under responsibility assignment strategy) for determining the most appropriate type of intentional element in a given situation (see Fig. 2), where the transition of one node to a child is based on a question (e.g., in node 1 at the tree at the left: "does the depender depend on the dependee to achieve an entity, or to attain a certain state? If entity, go to 3; if state, go to 2"). (b) The definition of a grammar for fixing the syntax of intentional elements in order to obtain uniform models from the point of view of naming (e.g., a Task's name is of the form: "Verb + (Object) + (Complement)", as in "Answer doubts by e-mail"). (c) The agreement on standard vocabularies for using as lexicon in the intentional models, e.g. the ISO/IEC 9126-1 standard [64] for quality concepts. On the other hand, Oliveira et al. [48] propose *i** Diagnoses, a method that uses questions as a way to elicit the intentional elements that compose the *i** models, e.g. "Why does <<dependee>> collaborate with <<depender>> to have <<goal>>?" and "What if <<goal>> is shared with another actor?". Both proposals claim that the methods produce more predictable models, although no validation supports these claims.

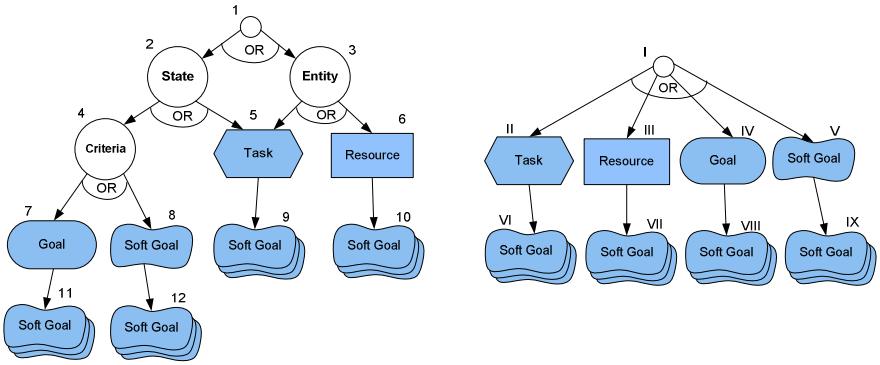

Fig. 2. Decision trees for determining the type of intentional element as defined in [22]

Grau et al. [32] propose the PR*i*M method framed in the business process reengineering problem. Detailed Interaction Scripts describe the current behaviour of the system in a scenario-like style. A set of prescriptive rules transform these scripts into *i** models that act as the basis for an activity of generation of alternatives to be evaluated as part of the reengineering process.

A comprehensive comparative analysis using 12 criteria and including Tropos, R*i*SD, PR*i*M and three other methods (GBM, ATM and BPD) may be found at [30].

Research directions. Given Yu's statement above, it is clear that aiming at designing fully deterministic *i** modelling methods should not be an ultimate goal. But based on the work described above, we may indicate some factors supporting the formulation of more prescriptive methods:

- *Steps*. The method shall consist of a series of well-defined steps and substeps. Remarkably, the model elements that may appear as input and output of each step and their relationships shall be defined in terms of the *i** metamodel.
- *Refinement rationale*. The method shall provide a clear rationale about: 1) whether is it still necessary to refine a given intentional element; 2) which kind of decomposition is needed; 3) which type do the decomposing intentional elements have. The use of questions as proposed in [22, 48] is probably the most comfortable way to proceed for the modeller.
- *Correctness checks*. The method shall provide verifiable means to check that the model being generated fulfils some identified conditions about their structure. The use of metrics [20] could help here.
- *Patterns*. The method shall contemplate the possibility of using knowledge patterns as a way to drive reusability. Patterns could be organized into different catalogues depending on the step where they apply (e.g., social patterns like in [27], but also requirement patterns as mentioned in [61, 70], design patterns, etc.).
- *Vocabulary*. The method shall promote the use of ontologies as a way to improve the accuracy of the models, as well as they consistence of different mod-

els over the same domain or system facet. Ontologies of interest may include domain ontologies like the REA enterprise ontology [66], or facet ontologies like the ISO/IEC 9126 quality standard [64] for non-functional characteristics used in soft goals. General knowledge of ontologies in the agent-oriented field [33] and proposals of representation of ontological concepts into the *i** framework [24] are worth to explore. On the other hand, the methods shall use a consistent grammar to name the intentional elements that compose the *i** models.

4 Challenge 3: Providing Structuring Mechanisms in *i**

Context. Modelling is a stepwise process. Some key elements are identified to build the starting model, and then a series of refinement steps gradually transforms this model into more concrete ones. In the *i** framework, the initial key elements usually are some designated actors (possibly with the main goal that they fulfil) and the most important dependencies among them. Refinement steps may yield to new actors and dependencies, and also to the gradual construction of the Strategic Rationale (SR) *i** diagram of each actor by decomposing their main goal using means-end and task-decomposition links, and establishing the contributions to softgoals.

Problem. This refinement process has not a clear counterpart in the *i** framework. The only structuring mechanism that *i** presents is the concept of actor boundary, that allows separating the declaration of existence of an actor from the rationale that it encloses. But the other refinement steps mentioned above are not supported by the language. Therefore, the final *i** model suffers from several problems:

- Difficult to reuse. If a model with some similarities has to be build in the future, reusability is basically copy and paste the designated elements, which is difficult and semantically poor. For instance, if a subpart of an SR diagram is a candidate to be reused, what happens to those dependencies that stem from its intentional elements?
- Difficult to trace. Since the model does not keep the stepwise refinement history, the reader is not able to know which elements were introduced in which stages and why. For an intentional framework like *i** is, this is even a more severe drawback because it hides some rationale.
- Difficult to understand. Since the model is a monolithic unit except for actor boundaries, the reader has more difficulties than ever to comprehend the full meaning of the system modelled.

Challenge. The *i** language shall include structuring mechanisms for representing the most usual stepwise refinement operations when developing *i** models.

State of the art. There are some lines of research addressing the structurability issue. The two most ambitious contributions at this respect are the incorporation of aspects and services into *i**.

Alencar et al. [3] propose the use of aspects for modelling cross-cutting concerns (see Fig. 3, left). Separation of concerns provides structure to the *i** models, but it does not align with the stepwise refinement process as presented above: identification of aspects and modularization of the model is made after the model has been written, therefore the development process is still not recorded. Also, the addition of aspects into *i** results in a framework with more modelling constructs and may eventually require a steeper learning curve.

Estrada incorporates the concept of service into the *i** framework [16] (see Fig. 3, right). This type of modularity unit is closer to the concepts managed in the domain (i.e., business services) and from this point of view fits better than aspects to the natural stepwise refinement process. However it is true that this particular proposal introduces a lot of complexity to the framework, with the fundamental concepts of "service" and "process", and also with the configuration of services inside SR boundaries using a variability-like model with mandatory and optional features combined in several ways. As in the approach above, validation is needed to assess usability of the proposal.

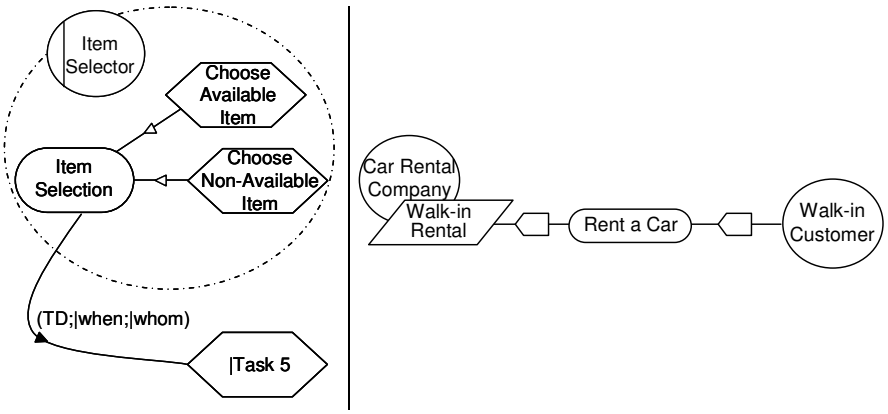

Fig. 3. Structuring *i** models using new constructs: aspects [3] (left) and services [16] (right)

If we consider proposals aligning with stepwise refinement, we still find several proposals. First, we mention the work by Leite et al. [40] that proposes a third kind of *i** diagram to complement the SD and SR diagrams, namely the SA (Strategic Actor) diagram that represents all kind of model actors (roles, positions and agents) with their relationships (plays, occupies, covers, instance, is-a, is-part-of), see Fig. 4, left, for an example. Also, Alencar et al. [1] introduce the concept of alternative for grouping means to achieve an end. This concept is generalized by Franch [21] into the general notion of module that is specialized into different types of SD- and SR-modules (see Fig. 4, right, for example). Remarkably, these three approaches define the introduced constructs in terms of metamodels.

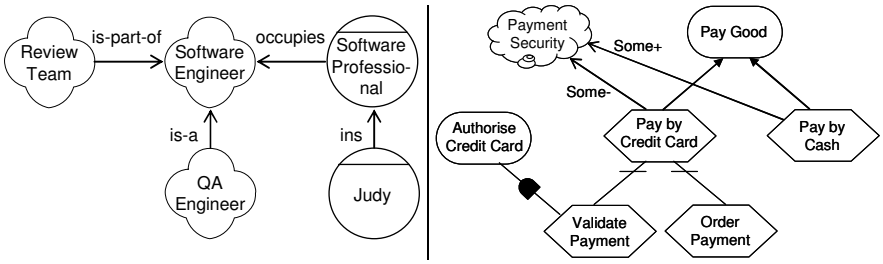

Fig. 4. Examples of structuring mechanisms: SA diagram [40] (left) and SR-module [21] (right)

Last, Lucena et al. [43] present a modularization approach that is built on the existing framework without any extension. This means using actors as only encapsulation mechanism. They provide a transformational framework in which transformation rules convert logically connected subgraphs of existing SR diagrams into SR diagrams for new actors.

Research directions. Basically we foresee two types of structuring mechanisms: domain-independent and domain-dependent.

- Domain-independent. The last three proposals fall into this first category and show a way to go. To sum up, the basic need is: 1) grouping dependencies that are related; 2) grouping related intentional elements inside an SR diagram; 3) defining submodels. Another related work could be analyzing if the aspect-oriented approach could be integrated with them, since both types of modularity constructs seem complementary. However, careful validation about usability of the resulting proposal should be conducted.
- Domain-dependent. The service concept as mentioned by Estrada is one example of concept that may be introduced. If we try to abstract from the service-oriented context to a general, open scope, we could think of introducing a generic structuring mechanism able to be instantiated by any kind of concept, in a way that this instantiation establishes: 1) the kind of *i** model elements that may take part of the module; 2) the relationships that need to be fulfilled.

In both cases, the following issues must be considered:

- Structuring mechanisms exist in virtually all modelling languages; therefore a systematic literature review is needed to learn how they do it, and to try to align to their principles. In particular, the analysis of the UML notion of package seems a must.
- The structuring mechanisms need to be introduced into the *i** metamodel in a non-intrusive way. Then structuring mechanisms become first-class citizens in the framework, whilst not interfering with the semantics of the intentional part.
- There is a need of defining the semantics of: 1) the modules themselves; 2) the module-combination operations (e.g., creation of a new module by the combination of existing ones); 3) the module application operation (i.e., the model that results from applying the stepwise refinement step implied by a module,

over an element of the departing model). As essential part of this issue, the classical model merging problem needs to be tackled [58].
- Tool-support is essential. Fundamental capabilities for applying the operations above, for managing a catalogue of modules and for providing views of the model based in the modules are needed.
- Visual representation. Being *i** a notation with a strong emphasis on the visual dimension, an informed decision about the visual representation of modules needs to be made. Work by Moody et al. [46] provides an excellent rationale for making this decision.

5 Challenge 4: Use *i** Models in Later Development Phases

Context. As an intentional modelling vehicle, the *i** framework is used in early stages of the system development. Some of the concepts that appear in *i** models will pervade in later stages, e.g. some *i** actors will act as such in use cases, some resources will appear also in conceptual data models, some tasks will become activities in a behaviour diagram, etc. This suggests for the need of having systematic ways to transform *i** models into other formalisms.

Problem. The transformation of goal models into more elaborate artefacts that appear later in the life-cycle has been tackled in several works (e.g., using the MAP approach to derive data-flow diagrams from goal models [51]). When trying to transform an *i** model into some other kind of model some difficulties arise. Typically the target of this transformation is a UML conceptual model [65], composed at least of a use case specification, a data conceptual model in the form of a class diagram, and perhaps some behavioural model. Sometimes just one of these artefacts is the target.

- Use cases are textual artefacts that reflect communication between actors in a sequential form. The problems that arise are: 1) identifying the appropriate use cases from the *i** model and also the relevant scenarios; 2) identifying the actors that take a part in each use case; 3) inferring the interactions between these actors and write them in the correct order; 4) generating the text itself.
- Data conceptual models are diagrams that include accurate and complete information about classes or entities, their relationships and their attributes. Discovering all of these elements from the *i** model is also a problem since the information that it encloses is not as complete as in data conceptual models (due to its intentional nature).
- Behavioural models like activity diagrams or sequence diagrams include interactions among actors, or activities to be performed, with a flow of control that is not expressed in *i** models.

Solving these problems can be considered a major challenge.

Challenge. There shall be techniques available to make easier (and automate up to a given extent) the transition from *i** models into other types of models.

State of the art. As mentioned above, the main research line related to this challenge corresponds to the transformation of *i** models into UML-like artefacts [50], generally in the context of model-driven development (MDD) [60]. Within the OO-Method MDD methodology [49], Alencar at al. have shown that it is possible to partially infer data conceptual models from *i** models [2]. Actors and their relationships, and resources (both dependencies and internal SR elements), play a fundamental role in this translation. Following the MDD foundations, transformation rules are defined to obtain an initial class diagram that is completed manually (e.g., adding information about multiplicity, not present at *i** models) for obtaining a complete OO-Method class model which can be used in the rest of the MDD process.

Concerning use case generation, Estrada et al. [17] propose a method that covers identification of use cases and actors, and writing of scenarios. Use cases are determined from both the task and resource dependencies that involve the actor that represents the system. The actors at the other end of such dependencies are represented as use case actors. Finally, SR diagrams are used to fill some predefined templates in order to generate the text of scenarios. A similar approach is followed by Santander and Castro [59].

Apart from these kind of models above, *i** has been used in other contexts. Remarkably, Ncube et al. [47] report an extension to the RESCUE process [38] in which a collection of 30 patterns were applied over an *i** model to generate textual candidate requirement statements using the VOLERE template, generating up to almost 600 requirements. As a result of this work, the authors argued that requirements generated from *i** models resulted in a more complete overall requirements specification.

Lucena et al. [42] have gone one step beyond in the development process and they address the generation of architectural models. They combine two levels of refinement, first by modularizing the departing model using the rules described in [43] and then transforming the resulting *i** model into a software architecture model described with the ACME architectural description language [25].

Research directions. As shown above, the transformation of *i** models into other models has been subject of much investigation. However, being a very complex topic, it requires still much work to do. When considering *i** models as the starting point of an MDD process, research is needed with respect to several topics [8]:

- Automating as much as possible the model transformation. It seems clear from previous work that full automation is not feasible since the underlying ontologies cannot be completely aligned. However, the work undertook so far (see above) looks promising and it may be expected that more results will be achieved soon. An important result of these approaches should be the clear statement of the limitations of the proposed methods regarding to automation. Also the possibility of enriching the *i** framework with information that in fact

belongs to the target ontology (e.g., order of task in task decompositions [23]) is a point to explore.
- Validating the adequacy of the *i** model before applying the transformations. Since the *i** model is not originally conceived for later transformation, it is necessary to assess its adequacy, e.g. how well-suited it is for generating classes and attributes in a class model. The definition and application of metrics using the *i*MDF$_M$ method [20] is a possible path to follow.
- Traceability among models. Traceability is a classical problem in MDD methods [52] and as such it needs to be properly managed.

6 Conclusion

In this chapter, we have argued the need of shifting the focus of the *i** community from pure research to a more practical view that may help in transferring the framework to practitioners. For the sake of brevity, we have focused on four challenges that have been described in detail, but several others are out there. These challenges solve some of the drawbacks that the empirical study by Estrada et al. has pointed out [18]. We have focused on scientific challenges, but there are also some other community-oriented issues worth to be considered, among them we may mention:

- Lessons learned. For putting *i** into practice, it is needed to have feedback about its use by practitioners. We think that the facts observed in the experiences mentioned at the introduction of this chapter should be consolidated by all the participants in these and others collaborative experiences, packaged into lessons learned, and the consequences fed back into the community as a main driver for identifying lines of future research.
- Tool support. In our opinion, taking into account the size of the core *i** research community, there is an unnecessary proliferation of tools. Given that developing and maintaining such tools has a considerable cost for the research groups, a possible strategic action could be to join efforts for producing a common subsystem at least for the more basic capabilities (e.g., *i** editor) configurable enough to adapt to each group's specificities, importing/exporting models in iStarML format and with a well-defined API, with a plug-in based infrastructure for enriching its functionality.
- Population for experiments. A great deal of current proposals of modelling variations, analysis techniques, development methods, etc., undergo through a weak validation (if any). The main reason for this probably is the difficulty on getting population enough to run these experiments. A community-oriented view is probably needed in order that the research groups allocate some of their effort in participating in such validations. The *i** wiki may help on implementing this idea.

We hope to see in the next years an increasing effort in these and other topics that make the use of the *i** framework in industrial cases not an exception but a usual practice.

Acknowledgments This work has been partially supported by the Spanish project TIN2007-64753.

References

1. Alencar F, Silva C, Lucena M, Castro J, Santos E, Ramos R (2008) Improving the Understandibility of *i** Models. In: Proc. of the 10th International Conference on Enterprise Information Systems, Volume ISAS-1, pp. 129-136
2. Alencar F, Marín B, Giachetti G, Pastor O, Castro J, Pimentel JH (2009) From *i** Requirements Models to Conceptual Models of a Model Driven Development Process. In: Proc. of PoEM 2009. LNBIP 39, pp. 99-114, Springer Berlin Heidelberg
3. Alencar F, Castro J, Lucena M, Santos E, Silva C, Araújo J, Moreira A (2010) Towards Modular *i** Models. In: Proc. of 25th Symposium of Applied Computing International Conference – RE Track (in press). ACM Press
4. Annosi A, de Pascale A, Gross D, Yu E (2008) Analyzing Knowledge Transfer in Software Maintenance Organizations using an Agent- and Goal-oriented Analysis Technique - an Experience Report. In: Proc. of 3rd International *i** Workshop, CEUR-WS 322, pp. 5-8
5. Annosi A, de Pascale A, Gross D, Yu E (2008) Analyzing Software Process Alignment with Organizational Business Strategies using an Agent- and Goal-oriented Analysis Technique - an Experience Report. In: Proc. of 3rd International *i** Workshop, CEUR-WS 322, pp. 9-12
6. Ayala C, Cares C, Carvallo JP, Grau G, Haya M, Salazar G, Franch X, Mayol E, Quer C (2005) A Comparative Analysis of *i**-based Agent-Oriented Modelling Languages. In: Proc. of 17th International Conference on Software Engineering and Knowledge Engineering Conference, pp. 43-50, Knowledge Systems Institute, Skokie
7. Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2004) Tropos: An Agent-oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems, 8(3): 203-236, Springer Berlin
8. Cabot J, Yu E (2008) Improving Requirements Specifications in Model-Driven Development Process. In: Proc. of International Workshop on Challenges in Model-Driven Software Engineering, available at http://ssel.vub.ac.be/ChaMDE08/wsorganisation, pp. 36-40
9. Cares C, Franch X, Mayol E, Quer C (2010) A Reference Model for *i**. In: Social Modelling for Requirements Engineering (in press). The MIT Press
10. Cares C, Franch X, Perini A, Susi A (2008) iStarML: An XML-based Model Interchange Format for *i**. In: Proc. of 3rd International *i** Workshop, CEUR-WS 322, pp. 13-16
11. Carvallo JP, Franch X (2009) On the use of *i** for Architecting Hybrid Systems: A Method and an Evaluation Report. In: Proc. of PoEM 2009. LNBIP 39, pp. 38-53, Springer Berlin Heidelberg
12. Carvallo JP, Franch X, Quer C (2008) Requirements Engineering for COTS-based Software Systems. In: Proc. of 23th ACM Symposium on Applied Computing, pp. 638-644, ACM, New York
13. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed Requirements Acquisition. Science of Computer Programming, 20(1-2): 3-50, Elsevier
14. Davies I, Green P, Rosemann M, Indulska M, Gallo S (2006) How Do Practitioners Use Conceptual Modelling in Practice? In: Data and Knowledge Engineering, 58: 358-380
15. DeLoach SA, Miller M (2010) A Goal Model for Adaptive Complex Systems. International Journal of Computational Intelligence: Theory and Practice, 5(2)

16. Estrada H (2008) A Service-oriented Approach for the *i** Framework. PhD Dissertation, Universidad Politécnica de Valencia
17. Estrada H, Martínez A, Pastor O (2003) Goal-Based Business Modelling Oriented towards Late Requirements Generation. In: Proc of ER 2003, LNCS 2813, pp. 277-290, Springer Berlin Heidelberg
18. Estrada H, Martínez A, Pastor O, Mylopoulos J (2006) An Empirical Evaluation of the *i** Framework in a Model-Based Software Engineering Environment. In: Proc. of CAiSE 2006, LNCS 4001, pp. 513-527, Springer Berlin Heidelberg
19. Franch X (2006) On the Quantitative Analysis of Agent-Oriented Models. In: Proc. of CAiSE 2006. LNCS 4001, pp. 495-509. Springer Berlin Heidelberg
20. Franch X (2009) A Method for the Definition of Metrics over *i** Models. In: Proc. of CAiSE 2009. LNCS 5565, pp. 201-215, Springer Berlin Heidelberg
21. Franch X (2010) Incorporating Modules into the i* Framework. In: Proc. of CAiSE 2010. LNCS (in press), Springer Berlin Heidelberg
22. Franch X, Grau G, Mayol E, Quer C, Ayala P, Cares C, Haya M, Navarrete F, Botella P (2007) Systematic Construction of *i** Strategic Dependency Models for Socio-Technical Systems. International Journal of Software Engineering and Knowledge Engineering, 17(1): 79-106, World Scientific Publishing Company, Singapore
23. Fuxman A, Kazhamiakin R, Pistore M, Roveri M (2003) Formal Tropos: Language and Semantics. Technical Report, http://dit.unitn.it/~ft/papers/ftsem03.pdf. University of Trento
24. Gailly F, España S, Poels G, Pastor O (2008) Integrating Business Domain Ontologies with Early Requirements Modelling. In: Proc. of RIGiM 2008. LNCS 5232, pp. 282-291, Springer Berlin Heidelberg
25. Garlan D, Monroe R, Wile D (1997) ACME: An Architecture Description Interchange Language. In: Proc. of Conference of the Centre for Advanced Studies on Collaborative Research (CASCON), pp. 7, IBM
26. Giachetti G, Alencar F, Franch X, Pastor O (2010) Applying *i** Metrics for the Integration of Goal-Oriented Modelling into MDD Processes. Technical Report ESSI-TR-10-2. Universitat Politècnica de Catalunya
27. Giorgini P, Kolp M, Mylopoulos J, Pistore M (2004) The Tropos Methodology: An Overview. In: Methodologies and Software Engineering for Agent Systems. Kluwer Academic Publishers
28. Giorgini P, Massacci F, Mylopoulos J, Zannone N (2005) Modelling Security Requirements through Ownership, Permission and Delegation. In: Proc. of 13th IEEE International Requirements Engineering Conference, pp. 167-176, IEEE CS Press, Los Alamitos
29. Giorgini P, Mylopoulos J, Nicciarelli E, Sebastiani R (2002) Formal Reasoning Techniques for Goal Models. In: LNCS 2503, pp. 167-181, Springer Berlin Heidelberg
30. Grau G, Cares C, Franch X, Navarrete F (2006) A Comparative Analysis of *i**Agent-Oriented Modelling Techniques. In: Proc. of 17th International Conference on Software Engineering and Knowledge Engineering Conference, pp. 657-663, Knowledge Systems Institute, Skokie
31. Grau G, Franch X (2007) On the Adequacy of *i** Models for Representing and Analyzing Software Architectures. In: Proc. of RIGiM 2007. LNCS 4802, pp. 296-305, Springer Berlin Heidelberg
32. Grau G, Franch X, Maiden NAM (2008) PR*i*M: An *i**-based Process Reengineering Method for Information Systems Specification. Information and Software Technology 50(1-2): 76-100
33. Guizzardi R, Guizzardi G, Perini A, Mylopoulos J (2006) Towards an Ontological Account of Agent-Oriented Goals. In: Proc. of SELMAS 2006. LNCS 4408, pp. 148-164, Springer Berlin Heidelberg
34. H*i*ME website, http://www.lsi.upc.edu/~llopez/hime/
35. Horkoff J, Yu E (2002) Evaluating Goal Achievement in Enterprise Modelling – An Interactive Procedure and Experiences. In: Proc. of PoEM 2009. LNBIP 39, pp. 145-160, Springer Berlin Heidelberg
36. *i** wiki, http://istar.rwth-aachen.de

37. ITU-T (International Telecommunication Union, Telecommunication Standardization Sector) (2008) Recommendation Z.151: User requirements notation (URN) - Language definition. http://www.itu.int/rec/T-REC-Z.151/e
38. Jones SV, Maiden NAM (2005) RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems. In: Mate JL, Silva A (eds.) Requirements Engineering for Socio-Technical Systems, pp. 245-265, Ideas Group
39. Kavakli E (2004) Modelling organizational goals: Analysis of current methods. In: Proc. 19th ACM Symposium on Applied Computing, pp. 1339-1343, ACM, New York
40. Leite J, Werneck V, de Pádua Albuquerque Oliveira A, Cappelli C, Cerqueira AL, de Souza Cunha H, González-Baixauli B (2007) Understanding the Strategic Actor Diagram: an Exercise of Meta Modelling. In: Proc. of 10th Workshop em Engenharia de Requisitos, pp. 2-12
41. Lockerbie J, Maiden NAM (2008) REDEPEND: Tool Support for *i** Modelling in Large-scale Industrial Projects. In: Proc. of CAiSE Forum, CEUR-WS 344, pp. 69-72
42. Lucena M, Castro J, Silva C, Alencar F, Santos E, Pimentel J (2009) A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models. In: Proc. of OTM 2009 Workshops, LNCS 5872, pp. 370-380, Springer Berlin Heidelberg
43. Lucena M, Silva C, Santos E, Alencar F, Castro J (2009) Modularizando Modelos *i**: uma Abordagem baseada em Transformação de Modelos. In: Proc. of 12th Workshop em Engenharia de Requisitos, pp. 33-44
44. Maiden NAM, Jones S, Ncube C, Lockerbie J (2010) Using *i** in Requirements Projects Some Experiences and Lessons Learned. In: Yu E, Giorgini P, Maiden NAM, Mylopoulos J (eds.) Social Modelling for Requirements Engineering (in press). The MIT Press
45. Mylopoulos J, Borgida M, Jarke M, Koubarakis M (1990) Telos: a Language for Managing Knowledge about Information Systems. ACM Transactions Information Systems 8(4): 327-362
46. Moody DL, Heymans P, Matulevicius R (2009) Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of *i** Visual Syntax. In: Proc. of 17th International Requirements Engineering Conference, pp. 171-180, IEEE CS Press, Los Alamitos
47. Ncube C, Lockerbie J, Maiden NAM (2007) Automatically Generating Requirements from *i** Models: A Case Study with a Complex Airport Operations System. In: Proc. of REFSQ 2007, LNCS 4542, pp. 33-47, Springer Berlin Heidelberg
48. Oliveira A, Leite J, Cysneiros L, Lucena C (2008) *i** Diagnoses: A Quality Process for Building *i** Models. In: Proc. of CAiSE Forum, CEUR-WS 344, pp. 9-12
49. Pastor O, Gómez J, Insfrán E, Pelechano V (2001) The OO-method Approach for Information Systems Modelling: from Object-Oriented Conceptual Modelling to Automated Programming. Information System, 26(7):507-534
50. Perini A, Susi A (2005) Automating Model Transformations in Agent-Oriented Modelling. In: Proc. of AOSE 2005, LNCS 3950, pp. 168-178, Springer Berlin Heidelberg
51. Prakash N, Rolland C (2006) System design for requirements expressed as a Map. In: Khosrow-Pour M (ed.) Emerging Trends and Challenges in Information Technology, pp. 501-503, Idea Group Inc Washington, USA
52. Ramesh B, Jarke M (2001) Toward Reference Models of Requirements Traceability. IEEE Transactions on Software Engineering 27(1): 58-93
53. Rolland C (2007) Capturing System Intentionality with Maps. In: Krogstie J, Opdahl AL, Brinkkemper S (eds.) Conceptual Modelling in Information Systems Engineering, pp. 141-158, Springer Berlin Heidelberg
54. Rolland C, Prakash N, Benjamen A (1999) A Multi-model View of Process Modelling. Requirements Engineering, 4(4): 169-187
55. Rolland C, Samia Kaabi R, Kraiem N (2007) On ISOA: Intentional Services Oriented Architecture. In: Proc. of CAiSE 2007, LNCS 4495, pp. 158-172, Springer Berlin Heidelberg
56. Rolland C, Souveyet C, Ben Achour C (1998) Guiding Goal Modeling Using Scenarios. IEEE Transactions on Software Engineering 24(12): 1055-1071

57. Roy JF, Kealey J, Amyot D (2007) Towards Integrated Tool Support for the User Requirements Notation. In: Proc. of SAM 2006. LNCS 4320, pp. 198-215, Springer Berlin Heidelberg
58. Sabetzadeh M, Easterbrook S (2006) View Merging in the Presence of Incompleteness and Inconsistency. Requirements Engineering Journal, 11(3):174-193, Springer
59. Santander V, Castro J (2002) Deriving Use Cases from Organizational Modelling. In: 10th International Requirements Engineering Conference, pp. 32-39, IEEE CS Press, Los Alamitos
60. Selic B (2003) The Pragmatics of Model-Driven Development. IEEE Software, 20(5):19-25, IEEE CS Press, Los Alamitos
61. Strohmaier M et al (2008) Can Patterns Improve *i** Modelling? Two Exploratory Studies. In: Proc. of REFSQ 2008, LNCS 5025, pp. 153-167, Springer Berlin Heidelberg
62. Susi A, Perini A, Mylopoulos J, Giorgini P (2005) The Tropos Metamodel and its Use. Informatica 29(4):401-408, Slovenian Society Informatika, Ljubljana
63. TAOM4E website, http://sra.itc.it/tools/taom4e/
64. The ISO Organization (2001) ISO/IEC Standard 9126 Software Engineering - Product Quality. ISO Standards
65. The OMG, http://www.uml.org/
66. The REA Ontology, https://www.msu.edu/user/mccarth4/rea-ontology/
67. University of Toronto, http://www.cs.toronto.edu/km/GRL/
68. van Lamsweerde A (2001) Goal-Oriented Requirements Engineering: A Guided Tour. In: Proc. of 5th IEEE International Symposium on Requirements Engineering, pp. 249, IEEE CS Press, Los Alamitos
69. Wachsmuth, G. (2007) Metamodel Adaptation and Model Co-adaptation. In: Proc. of ECOOP 2007. LNCS 4609, pp. 600-624, Springer Berlin Heidelberg
70. Yang J, Liu L (2008) Modelling Requirements Patterns with a Goal and PF Integrated Analysis Approach. In: Proc. of 32nd Annual IEEE International Computer Software and Applications Conference, pp. 239-246, IEEE CS Press, Los Alamitos
71. Yu E (1995) Modelling Strategic Relationships for Process Reengineering. PhD Dissertation, University of Toronto
72. Yu E (2009) Social Modelling in *i**. In: Conceptual Modelling: Foundations and Applications, LNCS 5600, pp. 99-121, Springer Berlin Heidelberg
73. Yu E, Giorgini P, Maiden NAM, Mylopoulos J (eds) (2010) Social Modelling for Requirements Engineering (in press). The MIT Press