iStar 2.0 Language Guide

Article · May 2016

3 authors:

Fabiano Dalpiaz
Utrecht University

146 PUBLICATIONS   **1,986** CITATIONS

Xavier Franch
Universitat Politècnica de Catalunya

422 PUBLICATIONS   **4,656** CITATIONS

Jennifer Horkoff
University of Gothenburg

109 PUBLICATIONS   **1,958** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   PLATEOSS View project

Project   Participatory Architectural Change MAnagement in ATM Systems (PACAS) View project

iStar 2.0 Language Guide

Fabiano Dalpiaz[1], Xavier Franch[2], and Jennifer Horkoff[3] *

[1] Utrecht University, The Netherlands
[2] Universitat Politècnica de Catalunya, Spain
[3] City University London, United Kingdom
`f.dalpiaz@uu.nl,franch@essi.upc.edu,horkoff@city.ac.uk`

Abstract. The *i** modeling language was introduced to fill the gap in the spectrum of conceptual modeling languages, focusing on the intentional (*why?*), social (*who?*), and strategic (*how? how else?*) dimensions. *i** has been applied in many areas, e.g., healthcare, security analysis, eCommerce. Although *i** has seen much academic application, the diversity of extensions and variations can make it difficult for novices to learn and use it in a consistent way. This document introduces the iStar 2.0 core language, evolving the basic concepts of *i** into a consistent and clear set of core concepts, upon which to build future work and to base goal-oriented teaching materials. This document was built from a set of discussions and input from various members of the *i** community. It is our intention to revisit, update and expand the document after collecting examples and concrete experiences with iStar 2.0.

Version History

Version	Date	Implemented Changes
3	June 17, 2016	New integrity rules: at most one actor link between a pair of actors; contribution and qualification cannot connect the same two elements.
2	June 3, 2016	Fixed typos in the original version
1	May 26, 2016	-

1   Motivation and Overview

The *i** language was presented in the mid-nineties [4] as a goal- and actor-oriented modeling and reasoning framework. It consists of a modeling language along with reasoning techniques for analyzing created models. *i** was quickly adopted by the research community in fields such as requirements engineering and business modeling. Benefiting from its intentionally open nature, multiple extensions of the *i** language have been proposed (see [2, 3] for useful reviews), either by slightly redefining some existing constructs, by detailing some semantic issues not completely defined in the seminal proposal, or by proposing new constructs for specific domains.

This flexible use of *i** has been fruitfully employed by researchers, who were able to benefit from a consolidated modeling and reasoning approach whilst tailoring it to their needs. However, this use has also some drawbacks. The most critical is the difficulty to spread the framework outside the experts' community:

- *Newcomers* find it hard to learn the intricacies of the language;
- *Educators* do not have a shared body of knowledge to teach;
- *Practitioners* are not provided with an established reference for using *i** in their projects;
- *Technology providers* cannot easily determine which are the core constructs to be implemented and the techniques to apply on top of those constructs.

As a response to the need of balancing the framework's open nature and a possible solution to the aforementioned adoption problems, the *i** research community started an initiative to identify a widely agreed upon set of core concepts in the *i** language. The main goal is to keep open the ability to tailor the framework while agreeing on the fundamental constructs.

This document summarizes the outcomes of the first iteration of the iStar[1] standardization process. To clearly distinguish this core language from its predecessors, we name it **iStar 2.0**.

The community discussed this language in several meetings and discussions starting in a dedicated one-day meeting before the ER'14 conference in Atlanta (October 2014). At the subsequent community meeting at CAiSE'15 in Stockholm (at the iStar teaching workshop, iStarT, June 2015), it was decided that a smaller group of researchers (the authors of this document) would guide the process, making concrete proposals and processing the inputs of the rest of the community. An initial draft of the core was discussed both at the iStar Workshop colocated with RE'15 (August 2015) and in another dedicated one-day meeting before ER'15 in Stockholm (October 2015). The obtained feedback has been incorporated into the document. A preliminary version was distributed among the researchers that participated in this process (December 2015), who provided a last round of comments, considered in this version (May 2016).

Each of the following five sections (Section 2–6) addresses a particular category of language constructs as presented in most *i** sources, for example, the iStar-wiki[2]. For each category, the document lists the concepts that are included in iStar 2.0, with a definition, necessary comments, concise examples and the

---

[1] The language is spelled iStar instead of *i** to allow better indexing through search engines.

[2] `http://istarwiki.org`

graphical representation. The focus of this document is on concepts and relationships; methodological possibilities are mentioned only briefly.

Several aspects are excluded from this first version of iStar but are planned for inclusion in the next versions. Among them, we mention the ontological definition of constructs (e.g., what is a goal?), visual representation (e.g., what is an effective graphical notation?), wording conventions (e.g, passive voices in goals), and methodological issues (e.g., when can a model be considered final?).

Illustrative example. We illustrate the concepts of iStar 2.0 using a running example concerning University travel reimbursement. Students must organize their travel (e.g., to conferences) and have several goals to achieve, and options to achieve them. To achieve their goals, students rely on other parties such as a Travel Agency and the university's trip management information system. We will introduce iStar concepts gradually, slowly building up the example. In Fig. 1 we show a final view of the example in order to give readers an early idea of the capabilities of iStar 2.0.
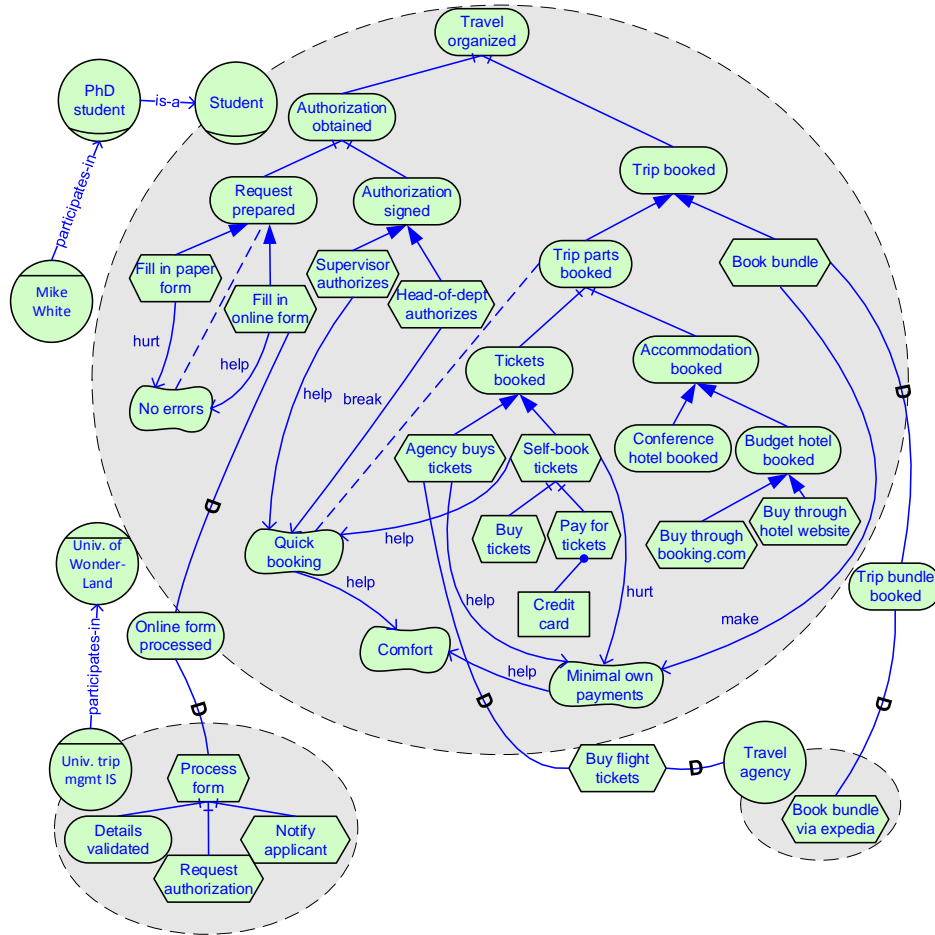

Fig. 1. A preview of the Travel Reimbursement Scenario as captured in iStar 2.0

Organization. Section 2 introduces the notion of actor and distinguishes between three types of actors. Section 3 presents the links in iStar 2.0 for relating actors. Section 4 describes the intentional elements that characterize the actors. Section 5 discusses the dependencies that socially relate the actors. Section 6 explains how the intentional elements can be linked. Section 7 details how to create different views of an iStar 2.0 model. Section 8 shows the metamodel of the language. Finally, we conclude and present future directions in Section 9.

2   Actors and actor types

Actors are central to the social modeling nature of the language. *Actors* are active, autonomous entities that aim at achieving their goals by exercising their know-how, in collaboration with other actors. In the iStar 2.0 language, two types of actors are distinguished:

– *Role*: an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor. Examples are: Student, PhD Student.
– *Agent*: an actor with concrete, physical manifestations, such as a human individual, an organization, or a department. Examples are: Travel agency, PhD Student, University of Wonderland, Mike White.

Whenever distinguishing the type of actor is not relevant, either because of the scenario-at-hand or the modeling stage, the notion of generic *actor*—without specialization—can be used in the model. For example, we can denote Travel agency as an actor to say that we do not know yet whether it is a specific agency (agent) or a characterization of the travel agency role.

Actors are represented graphically as circles. In the case of agent, a straight line is added in the top part of the actor circle. For a role, a curved line is added in the lower part. The graphical notation follows the mnemonic guidelines that the original *i** adopts[3]. Fig. 2 illustrates the notation.
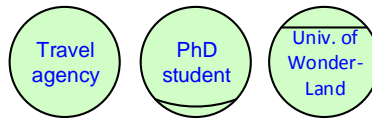

Fig. 2. Examples of actor, role and agent

Actors' intentionality is made explicit through the *actor boundary*, which is a graphical container for their intentional elements (see Section 4) together with their interrelationships (see Section 6). Fig. 3 shows the graphical representation of an actor boundary; elements and relationships will appear inside the grey area.

---

[3] These symbols are stylized depictions of a person wearing a hat and viewed from different angles. The Agent symbol is the frontal view where the name of the agent appears on the face. The Role symbol is an overhead view so that the label on the hat is visible.
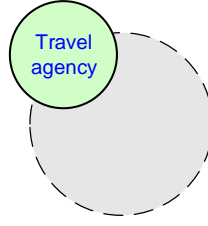
Fig. 3. Example of actor boundary

3   Actors association links

Actors are often interrelated. In iStar 2.0, this is captured via actor links that define/describe these relationships. Actor links are binary, linking a single actor to a single other actor. Two different types of actor links have been defined:

- *is-a*: represents the concept of generalization / specialization in iStar 2.0. Only roles can be specialized into roles, or general actors into general actors. For instance, a PhD student (role) can be defined as a specialization of a Student (another role). Agents cannot be specialized via is-a, as they are concrete instantiations (e.g., Mike White cannot be another agent).
- *participates-in*: represents any kind of association, other than generalization / specialization, between two actors. No restriction exists on the type of actors linked by this association. Depending on the connected elements, this link takes different meanings. Two typical situations are the following:
  - When the source is an agent and the target is a role, this represents the *plays* relationship, i.e., an agent plays a given role. For instance, Mike White plays the role of PhD student.
  - When the source and the target are of the same type, this will often represent the *part-of* relationship. For instance, the University trip management information system is part of the University of Wonderland.

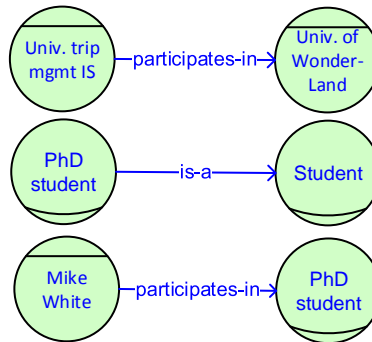 Every actor can *participate-in* multiple other actors.

Fig. 4. Examples of actor association links

Actor association links are represented using arrows in the diagram. The arrowhead identifies the target (the participated actor or the superclass, respectively).

A label identifying the type of link must be included. Fig. 4 shows the graphical representation of the three examples mentioned above.

4 Intentional elements

Intentional elements are the things actors want. As such, they model different kinds of requirements and are central to the iStar 2.0 language. An intentional element appearing inside the boundary of an actor denotes something that is desired or wanted by that actor. An intentional element can also appear outside of actor boundaries, as part of a dependency relationship between two actors (see Section 5). In this section, we focus on the former case: elements inside actor boundaries. The following elements are included in the language:

- *Goal*: a state of affairs that the actor wants to achieve and that has clear-cut criteria of achievement.
- *Quality*: an attribute for which an actor desires some level of achievement. For example, the entity could be the system under development and a quality its performance; another entity could be the business being analyzed and a quality the yearly profit. The level of achievement may be defined precisely or kept vague. Qualities can guide the search for ways of achieving goals, and also serve as criteria for evaluating alternative ways of achieving goals[4].
- *Task*: represents actions that an actor wants to be executed, usually with the purpose of achieving some goal.
- *Resource*: A physical or informational entity that the actor requires in order to perform a task.

Goals are graphically represented as ovals, while qualities are represented as more curved cloud-like shapes. Tasks are represented as hexagons to highlight their more structured definition in terms of a process to be followed. Resources are represented as rectangles. Fig. 5 shows examples of the intentional elements.

Fig. 5. Examples of intentional elements

5 Social dependencies

Dependencies represent social relationships in iStar 2.0. This, along with the assumption that actors can be human, organizations, technical systems (hardware, software), or any combination thereof, makes iStar 2.0 a socio-technical modeling language. A dependency is defined as a relationship with five arguments:

---

[4] iStar 2.0 departs from the original goal / softgoal dichotomy, which distinguishes these concepts based on the existence of a clear-cut metric for satisfaction, and from the Non-Functional/Functional Requirement distinction, as the use of this distinction varied in practice. By including qualities, which can be either "soft" or "hard" using *i** terminology, and by including the qualifies relationship between qualities and goals (Section 6.4), we clarify the relationships between goals and qualities.

- *depender* is the actor that depends for something (the dependum) to be provided;
- *dependerElmt* is the intentional element within the depender's actor boundary where the dependency starts from, which explains why the dependency exists;
- *dependum* is an intentional element that is the object of the dependency;
- *dependee* is the actor that should provide the dependum;
- *dependeeElmt* is the intentional element that explains how the dependee intends to provide the dependum.
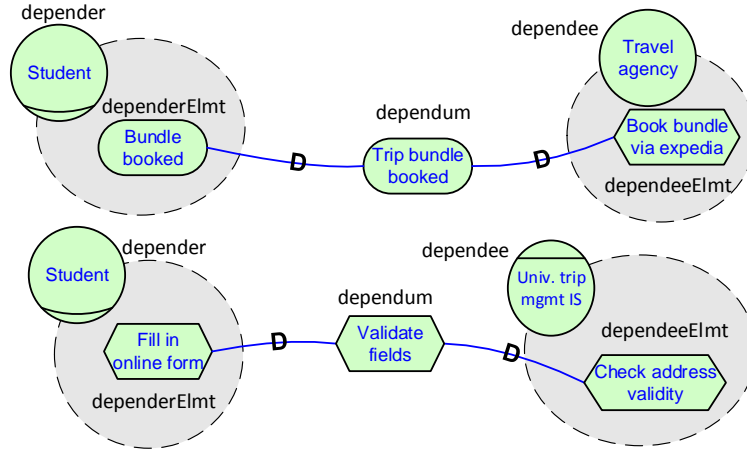

Fig. 6. Examples of dependencies

Dependencies (illustrated in Fig. 6) link the dependerElmt within the depender actor to the dependum, outside actor boundaries, to the dependeeElmt within the dependee actor. The link is drawn with a "D" symbol indicating direction, with the D acting as an arrowhead ">", pointing from dependerElmt to dependum to dependeeElmt.

Both the dependerElmt and the dependeeElmt can be omitted. This optionality is used when creating an initial Strategic Dependency view (see Section 7), or to support expressing partial knowledge, e.g., when the "why" (dependerElmt) or the "how"; (dependeeElmt) of the dependency are unknown. If both are omitted, the dependency links the depender actor to the dependee actor through the dependum. It is also possible to specify only one of them. See Fig. 7 for an example where the dependeeElmt is omitted.

The type of the dependum specializes the semantics of the relationship:

- Goal: the dependee is expected to achieve the goal, and is free to choose how;
- Quality: the dependee is expected to sufficiently satisfy the quality, and is free to choose how;
- Task: the dependee is expected to execute the task in a prescribed way;
- Resource: the dependee is expected to make the resource available to the depender.

This way, different dependency types indicate different degrees of freedom afforded by the depender to the dependee, with qualities and goals allowing the highest degree of freedom, tasks medium, and resource the lowest.
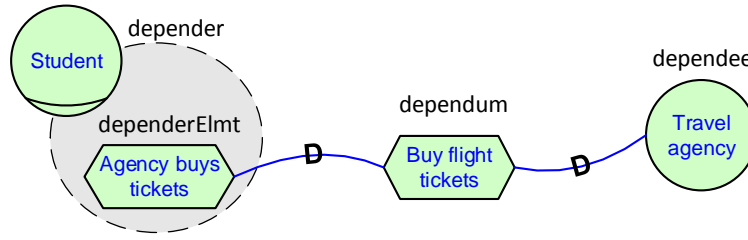
Fig. 7. Example dependency with dependeeElmt omitted

Rules and restrictions

When a depender depends on the dependee for its dependerElmt, the depender cannot or chooses not to satisfy/perform/have the dependerElmt on its own. Thus, the dependerElmt cannot be refined or contributed to.

It is possible that one or more of the dependerElmt, dependum, and dependeeElmt have the same element name. In this case, each of these elements is nevertheless distinct, as they reflect the separate viewpoints of the depender, the relationship between the two actors, and of the dependee respectively.

Dependency relationships should not share the same dependum, as each dependum is a conceptually different element; in some cases, a dependum in one dependency is achieved, but is not achieved in another dependency, even if the dependums may have the same name. In other words, an actor cannot depend on more than one actor for the same dependum, or two actors cannot depend on the same dependum from an actor.

6 Intentional element links

There are four types of links between intentional elements: *refinement*, *needed-by*, *contribution* and *qualification*. These are described in the following sub-sections and are summarized in Table 1.

Table 1. Links between intentional elements: overview

		Arrowhead pointing to			
		Goal	*Quality*	*Task*	*Resource*
Link starts from	*Goal*	Refinement	Contribution	Refinement	n/a
	Quality	Qualification	Contribution	Qualification	Qualification
	Task	Refinement	Contribution	Refinement	n/a
	Resource	n/a	Contribution	NeededBy	n/a

6.1 Refinement

To promote ease of adoption, iStar 2.0 features a generic relationship called *refinement* that links goals and tasks hierarchically. Refinement is an n-ary relationship relating one parent to one or more children. An intentional element can be the parent in at most one refinement relationship.

Two types of refinement exist—and apply to any kind of parent (goal or task)—that define the logical operator that relates the parent with the children:

- *AND*: the fulfillment of all the n children ($n \geq 2$) makes the parent fulfilled;
- *Inclusive OR*: the fulfillment of at least one child makes the parent fulfilled. This relationship allows for a single child.

A parent can only be AND-refined or OR-refined, not both simultaneously. Depending on the connected elements, refinement takes different meanings:

- If the parent is a *goal*:
    - In the case of AND, a *child goal* is a sub-state of affairs that is part of the parent goal, while a *child task* is a sub-task that must be fulfilled;
    - In the case of OR, a *child task* is a particular way (a "means") for fulfilling the parent goal (the "end"), while a *child goal* is a sub-goal that can be achieved for fulfilling the parent goal;
- If the parent is a *task*:
    - In the case of AND, a *child task* is a sub-task that is identified as part of the parent task, while a *child goal* is a goal that is uncovered by analyzing the parent task;
    - In the case of OR, a *child goal* is a goal whose existence that is uncovered by analyzing the parent task which may substitute for the original task, while a *child task* is a way to execute the parent task.

Refinement relationships do not imply a strictly top-down process, they can be built from the bottom-up, top-down, or via a mixed approach.

Graphically[5], *refinement* is expressed as a set of links directed from the sub-elements to the parent element. We employ a T-shaped arrowhead to denote AND-refinement, and a solid arrow directed towards the parent to represent OR-refinement (we use the original *i** symbol). See Fig. 8 for examples.
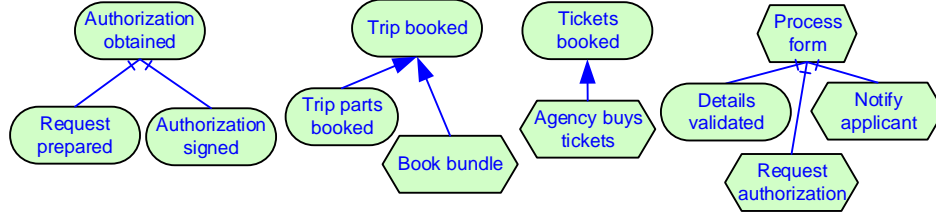

Fig. 8. Examples of refinement links

The first refinement shows a goal decomposed into two sub-goals that are both necessary (*AND-refinement*) to achieve the parent. The second refinement shows alternatives (*OR-refinement*): to book a trip, either the parts are booked, a bundle is booked, or both alternatives are chosen; while the former sub-element is a sub-state of affairs to achieve (a *goal*), the latter sub-element is a concrete set of actions to execute (a *task*). The third refinement exemplifies the existence of a single alternative. The fourth refinement shows the uncovering of goals while analyzing tasks: the goal "Details validated" is in the model because of the task "Process form".

---

[5] As mentioned in Section 1, an accurate study of the graphical definition of refinement (and other relationships) is left to future versions of the language.

6.2 NeededBy

The *NeededBy* relationship links a task with a resource and it indicates that the actor needs the resource in order to execute the task. This relationship does not specify what is the reason for this need: consumption, reading, modification, creation, etc. Graphically, NeededBy is represented as an arrow with a circle arrowhead directed towards the task, as shown in Fig. 9.
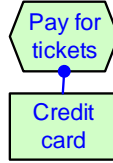

Fig. 9. Example of NeededBy relationship

6.3 Contribution

Contribution links represent the effects of intentional elements on qualities, and are essential to assist analysts in the decision-making process among alternative goals or tasks. Contribution links lead to the accumulation of evidence for qualities. We talk of qualities being *fulfilled* or *satisfied*, having sufficient positive evidence, or being *denied*, having strong negative evidence.

Contributions are defined as relationships from a source intentional element to a target quality, and having one of the following types:

- *Make*: The source provides sufficient positive evidence for the satisfaction of the target.
- *Help*: The source provides weak positive evidence for the satisfaction of the target.
- *Hurt*: The source provides weak evidence against the satisfaction (or for the denial) of the target.
- *Break*: The source provides sufficient evidence against the satisfaction (or for the denial) of the target.

Contributions are represented graphically as solid arrows with a text label that indicates the contribution type (see Fig. 10). While the examples show contributions starting from goals and tasks, it is also possible to initiate contributions from resources and qualities.
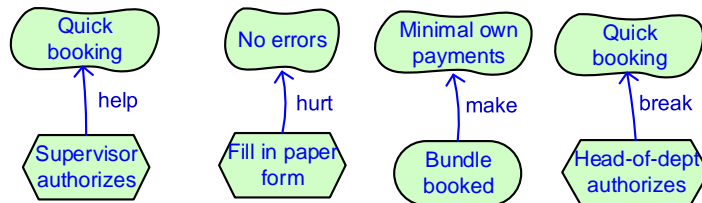

Fig. 10. Example of contribution links

6.4   Qualification

The *qualification* relationship relates a quality to its subject: a task, goal, or resource. For example, the quality "Quick booking" refers to the goal "Trip parts booked", it qualifies how the operation or function of this goal should be achieved. Similarly, the quality "No errors" refers to errors possibly created while fulfilling the goal "Request prepared", elaborating on how this goal might be achieved. Qualities are not necessarily attached to other elements through a qualification link. For example, "Avoid own payments" can be a standalone quality that does not qualify any other element, particularly if a task or goal concerning making one's own payments is not present in the model.

Placing a qualification relationship expresses a desired quality over the execution of a task, the achievement of the goal, or the provision of the resource. For example, in Fig. 11, saying that "No errors" qualifies "Request prepared" means that the preparation of a request should be achieved in such a way that it leads to no errors. The qualification relationship is represented graphically via a dotted line connecting the element that is qualifies.
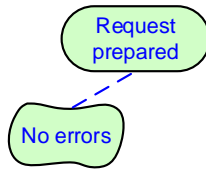
Fig. 11. Example of qualification link

7   Model views

When using iStar 2.0, the analyst creates a *model*. Such model can be visualized via multiple perspectives or *model views* (see also [1]). We specifically introduce three views that stem from the original *i** proposal and some extensions: the Strategic Rationale view, the Strategic Dependency view, and the Hybrid view.

Strategic Rationale (SR). The SR view shows all of the detail captured in the model, including actors, dependencies, actor association links, and the internal details of each actor. Modelers can view the strategic rationale inside each of the actors in the model. An SR view for the travel reimbursement scenario was shown at the beginning of this document, in Fig. 1.

Strategic Dependency (SD). The SD view shows each actor in the model, the actor association links, and the dependency relationships from depender to dependum to dependee. Fig. 12 shows an SD view of the travel reimbursement scenario.

Hybrid SD/SR. It is often useful to combine SD/SR views where some of the actors are open, but not all, focusing on the strategic rationale of a particular set of actors, and the actor links are hidden. This view is illustrated in Fig. 13.

Further useful views can be defined as needed, for example, the *actor view*, showing only actors and actor links, or a *functional view*, hiding all qualities, contribution and restriction links.
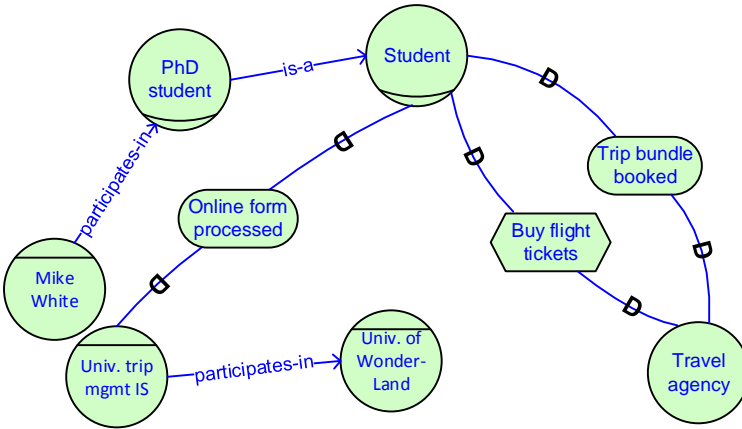
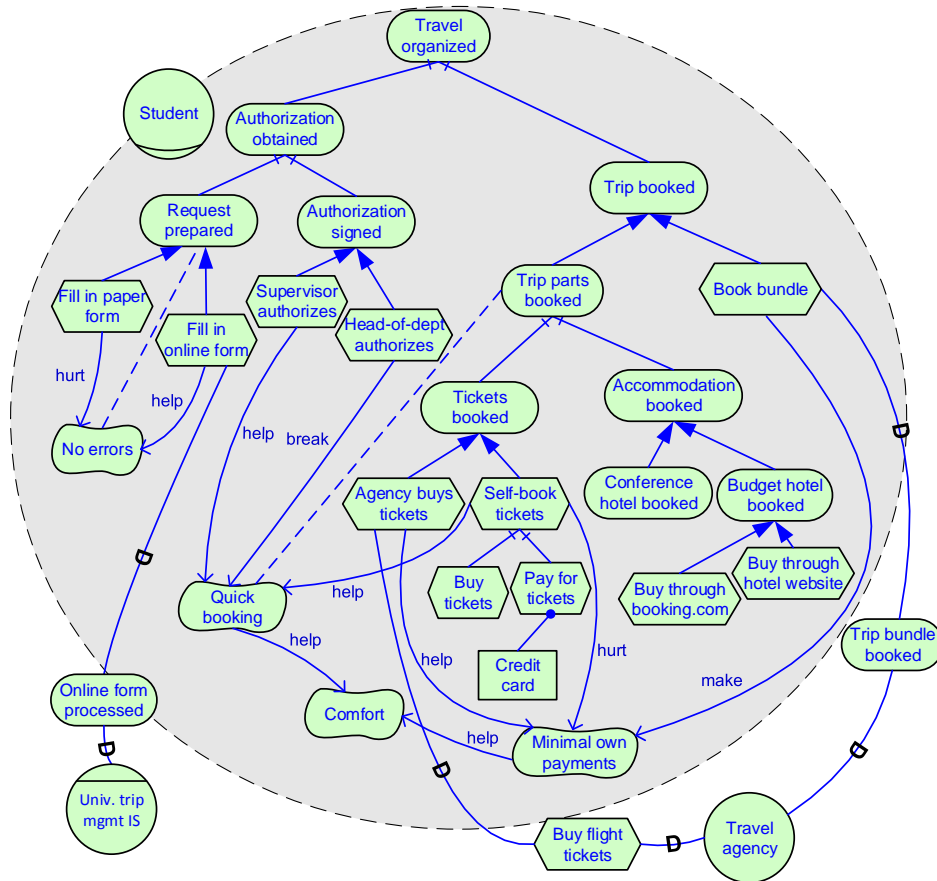**Fig. 12.** An SD view of the Travel Reimbursement scenario

Fig. 13. A hybrid SD/SR view of the Travel Reimbursement scenario

8   Metamodel

The metamodel for iStar 2.0 is shown in Fig. 14. The concepts and relationships in the metamodel have been explained and illustrated in the previous sections. We
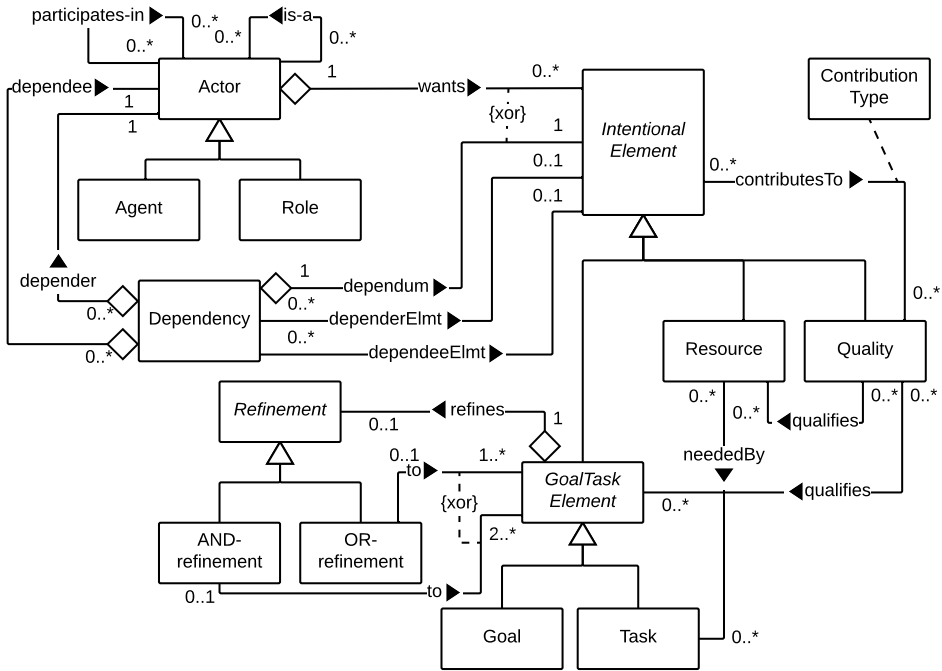
Fig. 14. Metamodel of iStar 2.0

describe a number of integrity constraints that explain more detailed constraints
on the models that the iStar 2.0 metamodel allows:

- The *is-a* relationship applies only between pairs of roles or pairs of actors;
- There should be no *is-a* cycles;
- There should be no *participates-in* cycles;
- A pair of actors can be linked by at most one actor link: it is not possible to
  connect two actors via both *is-a* and *participates-in*;
- In a dependency D, if the *dependerElmt* x exists, then the actor that wants
  x is the same actor that is D's *depender*;
- In a dependency D, if the *dependeeElmt* y exists, then the actor that wants
  y is the same actor that is D's *dependee*;
- The *depender* and *dependee* of a dependency should be different actors;
- For a dependency, if a *dependerElmt* x exists, then x cannot be refined or
  contributed to;
- The refinement relationship should not lead to refinement cycles (e.g., G OR-
  refined to G_1 and G_1 OR-refined to G, G OR-refined to G, etc.);
- The relationships between intentional elements (*contributesTo, qualifies, need-
  edBy, refines*) apply only to elements that are wanted by the same actor;
- An intentional element and a quality can be linked by either a *contributesTo*
  relationship or a *qualifies* relationship, but not by both;
- It is not possible for a quality to contribute to itself.

Finally, note that some classes in the metamodel are abstract (*GoalTask Element,
Intentional Element, Refinement*) as they are used to group together (via special-
ization) concrete classes that share some characteristics. On the other hand, *Actor*

is also specialized but it is a concrete class, thereby denoting that actors can be instantiated without further specialization.

9   Conclusion and Outlook

This document presented the results of the iStar standardization process that has led to the definition of the iStar 2.0 language. Supported by the research community of iStar, we promote the adoption of iStar 2.0 for educational and training purposes. To such extent, one of the next steps will be the creation of teaching materials that can be readily used to teach iStar 2.0.

This document does not conclude the standardization process. Following iterative design principles, we encourage the entire community to assist us in the conduction of studies about the ease of use, the adequacy for teaching, the expressiveness, the graphical notation, and the automated reasoning techniques that can support iStar 2.0.

Although our efforts in reconciling the numerous viewpoints of the community, we are well aware that there is still room for improvement. Therefore, we warmly welcome your feedback concerning the chosen primitives, the graphical notation, typographic errors, etc. We especially welcome examples of models that are created using iStar 2.0, which are especially helpful to pinpoint the aspects that can be improved.

Contact us by sending an e-mail or post public comments on the iStar 2.0 website: `https://sites.google.com/site/istarlanguage/`.

References

1. Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. *Security Requirements Engineering: Designing Secure Socio-Technical Systems*. MIT Press, 1 edition, 2016.
2. Jennifer Horkoff, Tong Li, Feng-Lin Li, Mattia Salnitri, Elsa Cardoso, Paolo Giorgini, John Mylopoulos, and João Pimentel. Taking goal models downstream: a systematic roadmap. In *International Conference on Research Challenges in Information Science*, pages 1–12. IEEE, 2014.
3. Jennifer Horkoff and Eric Yu. Comparison and evaluation of goal-oriented satisfaction analysis techniques. *Requirements Engineering*, 18(3):199–222, 2013.
4. Eric Siu-Kwong Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, 1996.