A Lightweight GRL Profile for i* Modeling

Daniel Amyot[1], Jennifer Horkoff[2], Daniel Gross[3], and Gunter Mussbacher[1]

[1] SITE, University of Ottawa
{damyot,gunterm}@site.uottawa.ca
[2] Department of Computer Science, University of Toronto
jenhork@cs.utoronto.ca
[3] Faculty of Information, University of Toronto
daniel.gross@utoronto.ca

Abstract. The i* framework is a popular conceptual modeling language for capturing and analyzing socio-technical motivation and properties of complex systems in terms of actors, their intentions, and their relationships. In November 2008, the International Telecommunications Union finalized the standardization of the User Requirements Notation (URN). URN is composed of two loosely coupled yet integrated sub-languages: the Goal-oriented Requirement Language (GRL), which is an intentional modeling language based on a subset of i*, and the Use Case Map notation for representing and capturing high-level system scenarios and structures. GRL was specifically defined in a non-restrictive way to foster the development and use of different agent and/or goal modeling approaches and techniques. However, because of its permissiveness, GRL can be used in ways that deviate from conventional i* modeling guidelines. In addition, some i* concepts do not have equivalent first-class concepts in GRL. In this paper, we present a lightweight GRL profile for i* that takes advantage of GRL's extensibility features to capture missing i* concepts. The profile presents formal constraints on the use of GRL and its extensions to restrict it to an i* style. Using GRL constrained by this profile enables GRL modeling and analysis tools to be used for i* models, and ensures that resulting i* models conform to an international standard and that they can be integrated with Use Case Maps. Variants and extensions of the original i* can also be supported in a similar way. This profile is implemented in the jUCMNav modeling tool.

Keywords: Goal-oriented Requirement Language, i*, jUCMNav, OCL, profile, User Requirements Notation.

1 Introduction

The i* modeling framework [12, 13] introduced aspects of intentional and social modeling and reasoning into information system engineering methods, especially at the requirements level. Unlike traditional systems analysis methods which strive to abstract away from the people aspects of systems, i* recognizes the primacy of social actors. Actors are viewed as being intentional, i.e., they have goals, beliefs, abilities, and commitments, which must be discovered, captured and analyzed. The analysis

C.A. Heuser and G. Pernul (Eds.): ER 2009 Workshops, LNCS 5833, pp. 254–264, 2009.

focuses on how well the goals of various actors are achieved given some configuration of relationships among human and system actors, and what reconfigurations of those relationships can help actors advance their strategic interests. Such analysis supports many software and system requirements engineering activities.

The i* framework has stimulated considerable interest in a socially-motivated approach to systems modeling and design, and has led to a number of extensions and adaptations, many of which are discussed in the i* Wiki [6]. One of these adaptations was recently standardized by the International Telecommunications Union (ITU-T) as part of the User Requirements Notation (URN – Recommendation Z.151) [7]. URN combines the Goal-oriented Requirement Language (GRL) with the Use Case Map (UCM) scenario notation in a single language, with a mature and well-defined metamodel supplemented by a concrete graphical syntax.

GRL supports many of the core concepts of i*, including actors, intentional elements, dependencies, contributions, and decompositions. However, GRL also differs from i* in a number of ways, such as the following:

1) *Missing first-class concepts in GRL*: i* contains concepts that are missing from GRL. For instance, GRL has only one type of actor, whereas i* also defines the notions of roles, agents and positions.
2) *GRL permissiveness*: GRL is voluntarily permissive in how intentional elements can be linked to each other. This is meant to support the wide variety of ways people actually create goal models [5]. However, i* proposes more specific and restrictive usages of relationships. For instance, an i* contribution link cannot have a task as a destination.
3) *Additional concepts in GRL*: GRL contains additional first-class concepts such as strategies (for the analysis of GRL models), metadata, and URN links (which enable the creation of typed links between any GRL/UCM elements).

In this paper, we present a lightweight profile for GRL that enables one to create goal models in a particular i* style according to the i* Guide in [6] and Yu's work [12, 13]. We take advantage of URN links and metadata to create relationships and stereotypes (respectively) for the missing GRL concepts found in i*. We specify constraints in UML's Object Constraint Language (OCL) [9] in order to restrict the usage of GRL to commonly used i* guidelines. We say that this profile is lightweight because it uses simple extensibility mechanisms and it does not require the extension of the URN metamodel or the use of heavyweight profiling mechanisms à la UML.

We also provide tool support for this profile with the jUCMNav tool, an Eclipse plug-in for the creation, analysis, and transformation of URN models [8, 10]. jUCMNav supports the notion of metadata together with an OCL engine that can check violations of user-defined constraints [2], enabling low-cost language customization.

A profile enabling the creation of GRL models in an i* style allows i* models to follow the standard defined in Z.151, including its interchange format. In addition, the use of the jUCMNav tool for i* models provides support for the division of models into consistent views (addressing scalability), the application of various pre-defined and automated quantitative and qualitative evaluation algorithms (with easy addition of new ones), the integration with UCMs, and simple modification or addition of constraints (for handling other variants and extensions of i*).

Because GRL is a recent language, background information on its notation and metamodel is given in section 2, followed by the profile definition in section 3. Section 4 presents the support of the profile in the jUCMNav tool. Related work is briefly discussed in section 5, followed by our conclusions.

2 Goal-Oriented Requirement Language (GRL)

GRL is a graphical language that focuses primarily on goal modeling. One of GRL's major assets is to provide ways to model and reason about functional and non-functional requirements in terms of goal achievement in a social context. With GRL, the modeler is primarily concerned with exposing "why" and "for whom" certain choices for behavior and/or structure were introduced, leaving the "what" and the "how" to other languages such as UCM and UML. GRL integrates core elements of i* [12, 13] and the NFR framework [3] relevant for intentional modeling. Major benefits of GRL over other popular notations include the integration of GRL with a scenario notation, the support for qualitative and quantitative attributes, and a clear separation of GRL model elements from their graphical representation, enabling a scalable and consistent representation of multiple views/diagrams of the same goal model.

The graphical syntax of GRL (see Fig. 1) is based on the syntax of the i* language. There are three main categories of concepts in GRL: actors, intentional elements, and links. A GRL goal graph is a connected graph of intentional elements that optionally reside within an actor boundary. An actor represents a stakeholder of the system or

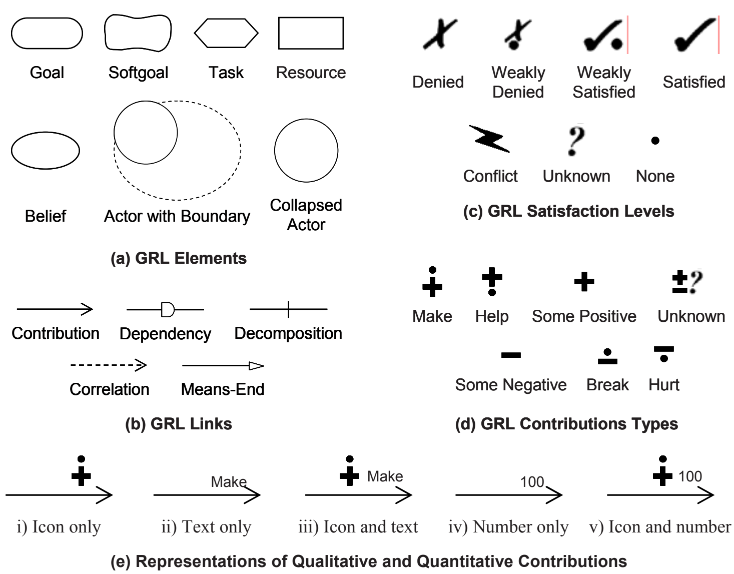

Fig. 1. Basic Elements of GRL Notation

another system. Actors are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available and softgoals to be satisfied. A goal graph shows the high-level business goals and system goals (functional and non-functional) of interest to a stakeholder and the alternatives for achieving these goals. A goal graph also documents beliefs (rationales) important to the stakeholder.

In addition to beliefs, *intentional elements* can be softgoals, goals, tasks, and resources. *Softgoals* differentiate themselves from *goals* in that there may not exist a clear, objective measure of satisfaction for a softgoal whereas a (hard) goal is usually quantifiable. In general, softgoals are more related to non-functional requirements, whereas goals are more related to functional requirements. *Tasks* represent solutions to (or operationalizations of) goals or softgoals. In order to be achieved or completed, softgoals, goals, and tasks may require *resources* to be available.

Links (see Fig. 1b) are used to connect elements in the goal model. *Decomposition links* allow an element to be decomposed into sub-elements. AND, IOR, as well as XOR decompositions are supported. XOR and IOR decomposition links may alternatively be displayed as *means-end* links. *Contribution links* indicate desired impacts of one element on another element. A contribution link can have a qualitative contribution type (see Fig. 1d), or a quantitative contribution (integer value between -100 and 100, see Fig. 1e). *Correlation links* are similar to contribution links, but describe side effects rather than desired impacts. Finally, *dependency links* model relationships between actors (one actor depending on another actor for something).

Fig. 2 presents the metamodel of the core GRL concepts, which constitute a part of the URN metamodel from Recommendation Z.151 [7]. These concepts represent the abstract grammar of the language, independently of the notation. This metamodel also formalizes the GRL concepts and constructs introduced earlier.

In addition, GRL inherits the concepts of *URN link* (Fig. 3-left), which enable one to create a link of a user-defined type between any pair of URN model elements (GRL

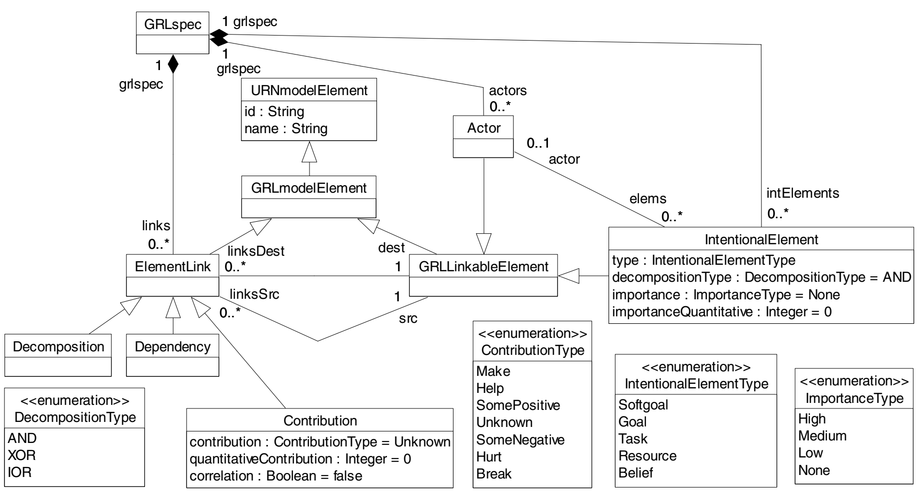

Fig. 2. GRL Metamodel – Core Concepts (from Z.151)

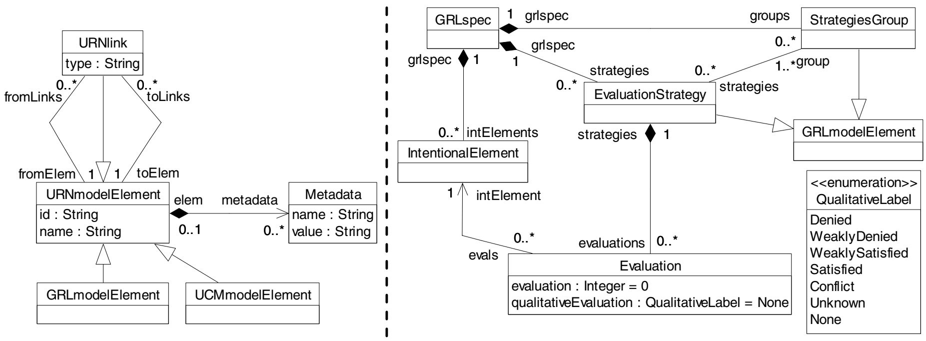

Fig. 3. GRL Metamodel – URN Links and GRL Strategies (from Z.151)

and UCM model elements alike). GRL model elements may also contain *metadata* that capture user-defined name-value pairs. These two concepts help extend the language or add precision to the model without having to change the metamodel.

Finally, a GRL model may also contain *evaluation strategies* (Fig. 3-right), which allow modelers to analyze the model for various what-if contexts. A strategy contains initial qualitative or quantitative evaluation levels attached to a subset of the intentional elements. These represent satisfaction levels that are propagated to the other intentional elements of the model through the decomposition, contribution, and dependency links. Several bottom-up evaluation algorithms (quantitative, qualitative and mixed) are proposed in the Appendix II of the URN standard [7] and are supported by jUCMNav [8, 10].

3 GRL Profile for i*

We have compared the i* Guide [6] with the URN standard [7] to determine what i* concepts were missing and where URN needs to be further constrained. This section presents a summary of the main differences and illustrates how they are supported in a lightweight profile.

3.1 Supplementary i* Concepts

i* supports many types of actors, namely *Role*, *Agent*, *Position*, and *Actor*. These can easily be supported using GRL actors (Fig. 2) to which we add metadata (Fig. 3-left) specifying the type of actor. This metadata element must have a name that indicates that this is actually a stereotype, e.g. `name="ST_iStar"`, and a value that specifies the type, e.g. `value="Role"`. A metadata name that starts with the `ST` prefix indicates a stereotype and the value will be displayed between « and » next to the element's name, e.g. `MyActorName «Role»`. This is similar in intent to UML's stereotypes.

There are also association links that exist between i* actors that are not covered by first-class GRL concepts, including: *ISA* (inheritance), *Is Part Of*, *Covers*, *Plays*, *Occupies*, and *INS* (instance of). These could be captured by a stereotype applied to a

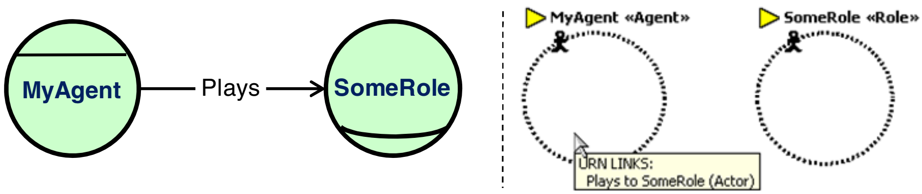

Fig. 4. Example: i* Actor Types and Association Links in GRL

dependency link, but this might be overstretching this concept. A better alternative is to use a URN link between the two actors (Fig. 3-left), where the link type corresponds to the desired i* association type. Note however that a drawback of URN links is that they do not have a visual representation in GRL diagrams (but these relationships can be exploited during analysis). The ►symbol indicates the presence of URN links on a GRL element, and tools like jUCMNav can show the nature of these from/to links in a tool tip, as shown in Fig. 4, where an agent plays some role.

As for i* *AND/OR* contribution links, they correspond semantically to GRL AND/OR decomposition links. The later should simply be used.

Finally, GRL does not make the distinction between *Strategic Dependency* (SD) models and *Strategic Rationale* (SR) models, as i* does. GRL has one integrated model, with multiple views (diagrams) based on the concrete syntax. URN provides additional metamodel elements to support the concrete syntax (location, size, colors, etc.), including one for GRL graphs (diagram). Hence, one can associate a stereotype to a GRL graph (ST_iStar metadata with value SD or SR) to enforce this distinction.

3.2 Constraints for i* Guidelines

Restricting the use of GRL to an i* style can be achieved by defining OCL constraints on the URN metamodel presented in section 2. Such constraints can target not only the core GRL concepts but also the various extensions described in section 3.1. Rules presented here use the conventions proposed in [5] and are tagged as strict (must never be violated) or loose (should not happen, but is tolerable).

One commonly accepted rule for i* is the following: ***(Strict)*** *Contribution links must only have softgoals as destinations*. In terms of the URN metamodel, this means that the dest GRLLinkableElement of an ElementLink which is a Contribution must be an IntentionalElement with type Softgoal. The corresponding OCL invariant is:

```
context Contribution
inv SoftgoalAsContributionDestination:
    self.dest.oclIsTypeOf(IntentionalElement)
    implies
    (self.dest.oclAsType(IntentionalElement)).type =
        IntentionalElementType::Softgoal)
```

Other i* constraints targeting intentional elements and links include the following (OCL not included for brevity):

- **(Strict)** Decomposition links must not have softgoals, resources or beliefs as a destination. In GRL terms, an ElementLink which is a Decomposition must not have a dest IntentionalElement with type Softgoal, Resource or Belief.
- **(Strict)** Decomposition links must not have beliefs as a source.
- **(Loose)** Beliefs should not be the destination of element links.
- **(Loose)** AND decomposition links should only have tasks as destinations.
- **(Loose)** Means-end links (i.e., OR/IOR decomposition links in GRL) should only have goals as destinations.

Interestingly, one can also define constraints that involve the metadata/stereotypes and URN links used to add i* concepts to GRL, as explained in the previous section. For instance, the rule ***(Strict)** ISA (generalization) must be between two actors of the same type* can be encoded in OCL using the following constraint:

```
context Actor
inv ISAbetweenActorsOfSameType:
    self.getLinksTo('ISA')->
        forall(to | to.oclIsTypeOf(Actor) and
           ( to.oclAsType(Actor).getMetadata('ST_iStar') =
             self.getMetadata('ST_iStar') )
        )
```

The above rule states that for all the URN elements that are the targets of URN links of type ISA, each such element (`to`) must be an Actor and must have the same ST_iStar metadata value as the source Actor. The rule takes advantage of two reusable OCL helper functions defined in our framework to query URN links (`getLinksTo`) and metadata (`getMetadata`). Other rules involving metadata and URN links include:

- (**Loose**) An *Is Part Of* association should be between two actors of the same type.
- (**Strict**) A *Covers* association must be from a Position to a Role.
- (**Strict**) A *Plays* association must be from an Agent to a Role.
- (**Strict**) An *Occupies* association must be from an Agent to a Position.
- (**Strict**) An *INS* association must only be used between Agents.

Finally, similar restrictions can also target actor boundaries and dependencies (note that the other relevant situations are already covered by standard GRL constraints).

- **(Strict)** Dependency links must never completely be inside of an actor boundary. In GRL terms: For an ElementLink which is a Dependency (with source and destination GRLLinkableElements):
  - If src and dest are both Actors, then dest ≠ src
  - If src is an Actor and dest an IntentionalElement, then src ≠ dest.actor
  - If dest is an Actor and src an IntentionalElement, then src.actor ≠ dest
  - If src and dest are both IntentionalElement, then src.actor ≠ dest.actor

- (**Strict**) Dependency links in an SD model must always have a dependum, i.e., there should never be a dependency link from an actor to an actor.
- (**Strict**) SD models must not have links other than dependency and actor association links.
- (**Loose**) Dependency links in an SR model should always have a dependum.
- (**Loose**) The only links that cross actor boundaries should be dependency links.

4 Tool Support

The lightweight GRL profile for i* was implemented in the jUCMNav tool. A GUI for managing metadata and URN links is already available [10], so supporting the supplementary i* concepts from section 3.1 and Fig. 4 is already covered.

The integrated OCL-based engine for the verification of user-defined semantic rules presented [2] can also be used as is to define and check the i* constraints highlighted in section 3.2. The `SoftgoalAsContributionDestination` rule previously see is repeated in Fig. 5. The name, context, and constraint expression are essentially the same, except for the precision of metamodel packages (`grl::`) required by jUCMNav. The tool also allow for the definition of an informal description and of supplementary utility functions (such as `getLinksTo` and `getMetadata`). An OCL query expression is required to collect all the instances of a particular URN metaclass (`Contribution` here) used in the model being edited. Such a rule is created once and can then be checked against any URN model. The tool also allows for rules to be exported and imported, so modelers can share their rules.

Constraints to be checked can be selected individually. For convenience, constraints can also be grouped to ease their selection. In Fig. 6, several groups of constraints are present, including three groups related to our profile for i*: one for strict rules, one for loose rules, and one for both (as a constraint can be part of multiple groups).

The verification of constraints is done on demand via menu selection. Violations are reported in Eclipse's standard Problems view. For example, suppose an i* model

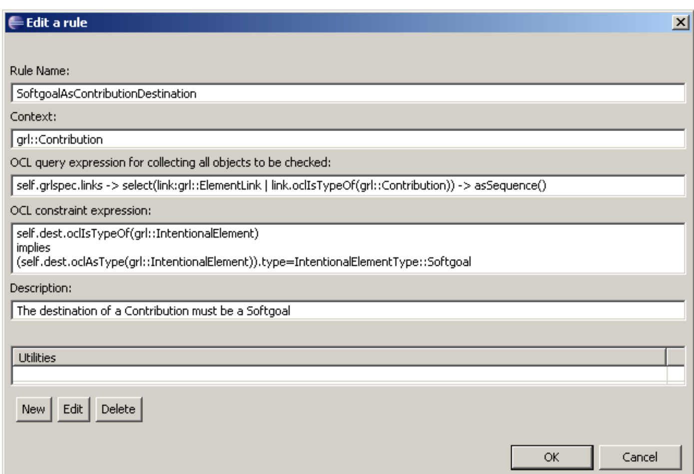

Fig. 5. Example of Constraint Definition: *SoftgoalAsContributionDestination*

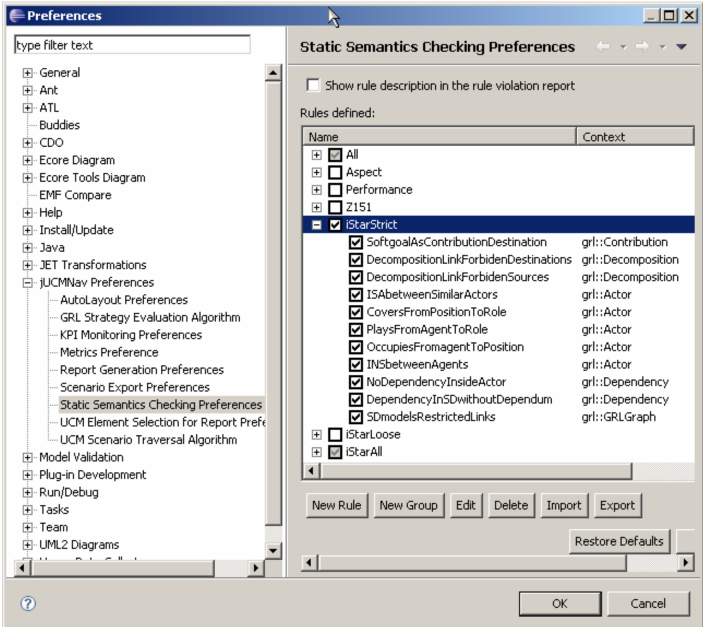

Fig. 6. Selectable Strict and Loose i* Constraints in the GRL Profile

where one of the tasks contributes to a goal, a situation that is forbidden according to the `SoftgoalAsContributionDestination` constraint. Out of the 11 strict i* constraints selected in Fig. 6 and checked against the model, one error would be detected and reported in the Problems view. Double-clicking on the error actually brings the focus of the model editor to the violating element (the contribution in this case).

5 Related Work and Discussion

General UML profiling for goal modeling was explored by Supakkul and Chung [11], with integration to Use Case diagrams. Their implementation is however not focused on i*, and constraints such as those described in our work are not checked. Grangel *et al.* [4] also introduced a metamodel-based UML profile, with support in a commercial tool (IBM RSM). Still, our goal metamodel is more general and standard than the one they used, which is specialized for enterprise goals. Abid created a UML profile for GRL [1] and also integrated it to a commercial tool (Telelogic Tau). This is however a heavyweight profile which allows neither the checking of constraints nor the exploration of i* variants. The i* Wiki [6] reports on many tools for i* modeling, but none is using a standard format and none allows user-selectable constraints to be checked. Our approach, although illustrated here with the i* constraints documented in [5], could be applicable to other i* variants (including TROPOS) and favorite styles.

One important benefit of such profile is that the evaluation features of GRL and jUCMNav (see Fig. 3-right) become available for i* models. In addition, jUCMNav allows one to add new evaluation algorithms [10], some of which could take advantage of the new stereotype and URN link information now captured. This would also permit one to compare evaluation procedures from i* and GRL. In addition to the support of UCM scenarios, jUCMNav also contains extensions of GRL not included in standard URN (e.g. for key performance indicators and aspect-oriented modeling), which are now available as a byproduct to i* models for further exploration.

6 Conclusions

We have introduced a lightweight profile for GRL that allows the representation of concepts unique to i* and restricts the usage of GRL to comply with i* guidelines. This is supported by the jUCMNav tool, where constraints can be captured in OCL and violations reported to modeler. This work casts i* onto a standardized metamodel (URN's) that enables the use of GRL-like analysis for i* models, the comparison of i* stylistic guidelines and evaluation algorithms, and a common representation of i* models. Using the underlying mechanisms presented here (metadata and URN links), language designers can prototype and explore new language features easily and at low cost before making them first-class entities in a revised metamodel.

Several directions for future work were already identified in the previous section. In addition to those, the jUCMNav tool could be improved in a number of ways, including by reporting warnings (instead of errors) for violating constraints that are loose (instead of strict). Also, more in-depth, hopefully industrial case studies to test the use of i* models with jUCMNav and GRL evaluation is required.

Acknowledgments. This work was supported by NSERC (Discovery Grants and Postgraduate Scholarships). We also thank Eric Yu for his helpful comments.

References

1. Abid, M.R.: UML Profile for Goal-oriented Modelling. Master of Computer Science Thesis, University of Ottawa, Canada (2008)
2. Amyot, D., Yan, J.B.: Flexible Verification of User-Defined Semantic Constraints in Modelling Tools. In: CASCON 2008. ACM, New York (2008)
3. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
4. Grangel, R., Chalmeta, R., Campos, C., Sommar, R., Bourey, J.-P.: A Proposal for Goal Modelling Using a UML Profile. In: Enterprise Interoperability III, pp. 679–690. Springer, Heidelberg (2008)
5. Horkoff, J., Elahi, G., Abdulhadi, S., Yu, E.: Reflective Analysis of the Syntax and Semantics of the i* Framework. In: Song, I.-Y., Piattini, M., Chen, Y.-P.P., Hartmann, S., Grandi, F., Trujillo, J., Opdahl, A.L., Ferri, F., Grifoni, P., Caschera, M.C., Rolland, C., Woo, C., Salinesi, C., Zimányi, E., Claramunt, C., Frasincar, F., Houben, G.-J., Thiran, P. (eds.) ER Workshops 2008. LNCS, vol. 5232, pp. 249–260. Springer, Heidelberg (2008)
6. i* wiki, http://istar.rwth-aachen.de (accessed, May 2009)
7. ITU-T – International Telecommunications Union: Recommendation Z.151 (11/08) User Requirements Notation (URN) – Language definition. Geneva, Switzerland (2008)

8. jUCMNav website. University of Ottawa, http://jucmnav.softwareengineering.ca/jucmnav/ (accessed, May 2009)
9. OMG – Object Management Group: Object Constraint Language Specification, 2.0 (2006)
10. Roy, J.-F., Kealey, J., Amyot, D.: Towards Integrated Tool Support for the User Requirements Notation. In: Gotzhein, R., Reed, R. (eds.) SAM 2006. LNCS, vol. 4320, pp. 198–215. Springer, Heidelberg (2006)
11. Supakkul, S., Chung, L.: A UML profile for goal-oriented and use case-driven representation of NFRs and FRs. In: Dosch, W., Lee, R.Y., Wu, C. (eds.) SERA 2005. LNCS, vol. 3647, pp. 29–41. Springer, Heidelberg (2006)
12. Yu, E.S.K.: Modelling strategic relationships for process reengineering. Ph.D. dissertation. Dept. of Computer Science, University of Toronto, Canada (1995)
13. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE Int. Symp. on RE, Washington, USA, pp. 226–235. IEEE CS, Los Alamitos (1997)