A UML Profile for Goal-Oriented Modeling

Muhammad R. Abid, Daniel Amyot, Stéphane S. Somé, and Gunter Mussbacher

SITE, University of Ottawa, 800 King Edward, Ottawa, ON, K1N 6N5, Canada
{mabid006, damyot, ssome, gunterm}@site.uottawa.ca

Abstract. The Unified Modeling Language (UML) does not fully address the needs of some important modeling domains, including goals and non-functional requirements (NFR). However, UML can be extended and tailored through the definition of profiles. In this paper, we propose a UML profile for the Goal-oriented Requirement Language (GRL), the goal/NFR notation of the User Requirements Notation (URN), recently standardized by ITU-T as Recommendation Z.151. Our profile is based on the abstract metamodel of GRL defined in accordance with ITU-T Recommendation Z.111 (meta-metamodel). This GRL metamodel has already been successfully tested and implemented in the jUCMNav Eclipse plug-in (a URN modeling tool). The profiling approach used in this paper adheres to the guidelines for UML profile design defined in ITU-T Recommendation Z.119. The resulting profile has been implemented in a UML 2 tool, namely IBM Rational Tau 4.0, and validated with case studies. Our experience and lessons learned are also discussed.

Key words: Goal-oriented Requirement Language, Metamodel, Tau G2, UML Profile, User Requirements Notation

1 Introduction

Goals are high-level objectives or concerns of a business, stakeholder, or system. They are often used to discover, select, evaluate, and justify requirements for a system. *Functional Requirements* (FR) define functions of the system under development, whereas *Non-Functional Requirements* (NFR) characterize system properties and qualities such as expected performance, robustness, usability and cost. Goals and NFRs capture essential aspects of systems, which have a significant impact throughout the development process. Goal models help to compare alternative requirements and solutions, facilitate trade-offs among conflicting concerns and constraints of different stakeholders, document rationales for design decisions (so we avoid revisiting previously considered and inferior solutions), and provide traceable justifications for lower-level requirements and design artifacts [4, 19].

The Unified Modeling Language (UML) [16] is the most popular modeling language for developing software applications. However, many modelers are still unsatisfied with the role of UML in the area of goal and NFR modeling. With the

importance of UML in the industry, this deficiency has now become an apparent weakness.

Modelers struggle to define how best to describe and structure goals. While some metamodels for goal modeling languages exist, such work has often been completed in isolation and has not been done in accordance with standards. Yet, there exists one mature metamodel for goal modeling, namely that of the Goal-oriented Requirement Language (GRL). GRL is combined with the Use Case Map (UCM) scenario notation to form the User Requirements Notation (URN) [11], recently standardized as ITU-T Recommendation Z.151 [12]. Standalone GRL-based modeling is useful on its own, and even more so when combined with UCMs. However, if one desires to integrate goal modeling with UML, then having a standardized and standalone goal metamodel may not be sufficient, and aligning it with the UML metamodel would help reduce existing communication and integration problems between goal modelers and UML modelers.

Although UML does not address explicitly the modeling of goals and non-functional requirements, there is a generic extension mechanism for tailoring UML to a particular domain, namely *UML profiling*. In this paper, we propose a UML profile for GRL. This profile satisfies the following properties, which are considerably important for successful goal-oriented modeling in a UML context.

1. *Integration with UML*: the ability to share information between the goal model elements and other UML elements.
2. *Diagram pollution avoidance*: preventing the mixing of different diagram constructs.
3. *Metamodel stability*: the maturity of the underlying goal metamodel.
4. *Implementability of the profiling mechanism*: how well the approach used for the creation of the profile is amenable to implementation and tool support.

A good integration with UML allows one to maintain traceability between goal models and UML models. The ability to avoid diagram pollution is useful to ensure the consistency of the goal diagrams created, and this is typically achieved by providing a dedicated diagram editor. A mature and stable metamodel contributes to the resulting profile's stability and validity. Finally, the implementability of a profile is important for integrating usable editors in existing UML modeling environments.

In our work, we satisfy the first property by virtue of UML profiling. Diagram pollution avoidance results from our choice of metamodeling implementation approach (*metamodel extension*, to be detailed in Sect. 3). Because our profile is based on a standardized metamodel (URN's, explored in Sect. 2), which has been already used as basis for GRL modeling tools [14], our work satisfies the stability property. The implementability of the profiling mechanism is demonstrated by a proof-of-concept implementation based on a commercial UML tool, to be described in Sect. 4.

Our main contributions are hence the definition of a UML profile for goal modeling rooted in the standard GRL metamodel and its implementation in

IBM Rational (previously Telelogic) Tau 4.0 [8]. UML's metaclasses are mapped to GRL's metaclasses in accordance with standard guidelines provided in ITU-T Recommendation Z.119 [10]. We also discuss the typical usage of this profile with a small example where GRL is used standalone in a model, and then where GRL diagrams are combined with selected UML diagrams in a model. Related work is discussed in Sect. 5, followed by our conclusions.

2 Goal-oriented Requirement Language (GRL)

This section introduces the GRL notation, main concepts, and metamodel.

2.1 Overview of the GRL Notation

GRL is a language that focuses primarily on goal modeling. A subset of URN and a graphical language, GRL's major asset is to provide ways to model and reason about non-functional requirements and other high-level goals. With GRL, the modeler is primarily concerned with exposing "why" 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 some of the best elements of two well-known goal-oriented modeling languages, i* [20] and the NFR framework [4]. 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 [20]. 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 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 non-functional requirements of interest to a stakeholder and the alternatives for achieving these high-level elements. 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 is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable. In general, softgoals are related more to non-functional requirements, whereas goals are related more 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. 1.b) are used to connect elements in the goal model. *Decomposition links* allow an element to be decomposed into sub-elements. AND, IOR, as

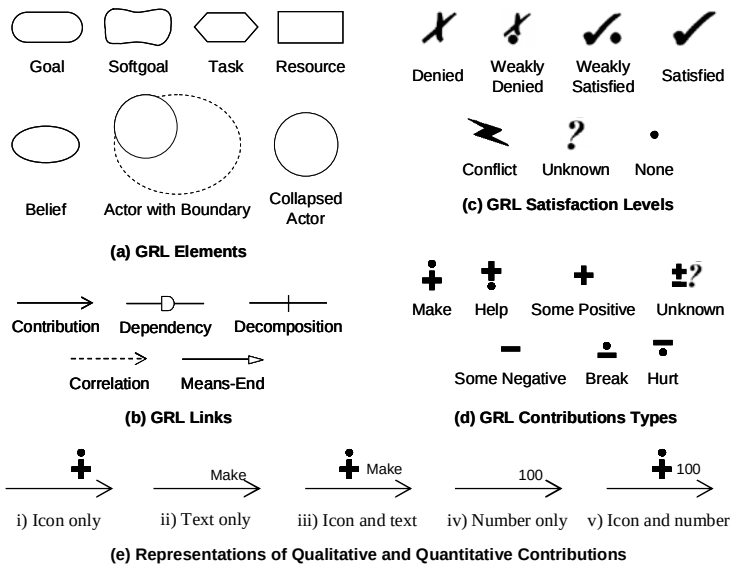

Fig. 1. Basic Elements of GRL Notation

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. 1.d), or a quantitative contribution (integer value between -100 and 100, see Fig. 1.e). *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).

As an example, Fig. 2 shows a simple GRL model that illustrates the use of GRL constructs. This model captures various relationships between the concerns of customers, the bank and the merchant involved in some transaction payment. This is an artificial example used for testing our profile as it covers many combinations of links (including contributions with various weights), actors and intentional elements of different types in a single diagram. It will not be explained further here as it is not meant to reflect reality (for a GRL tutorials and literature, please refer to [3]).

2.2 GRL Metamodel

Figure 3 shows a graphical representation of the metamodel of the core GRL concepts, which constitute a part of the URN metamodel from Recommendation Z.151 [12]. These concepts represent the abstract grammar of the language, independently of the notation.

This metamodel formalizes all the GRL concepts and constructs introduced earlier. The GRL metamodel definition adheres to the guidelines of ITU-T Rec-

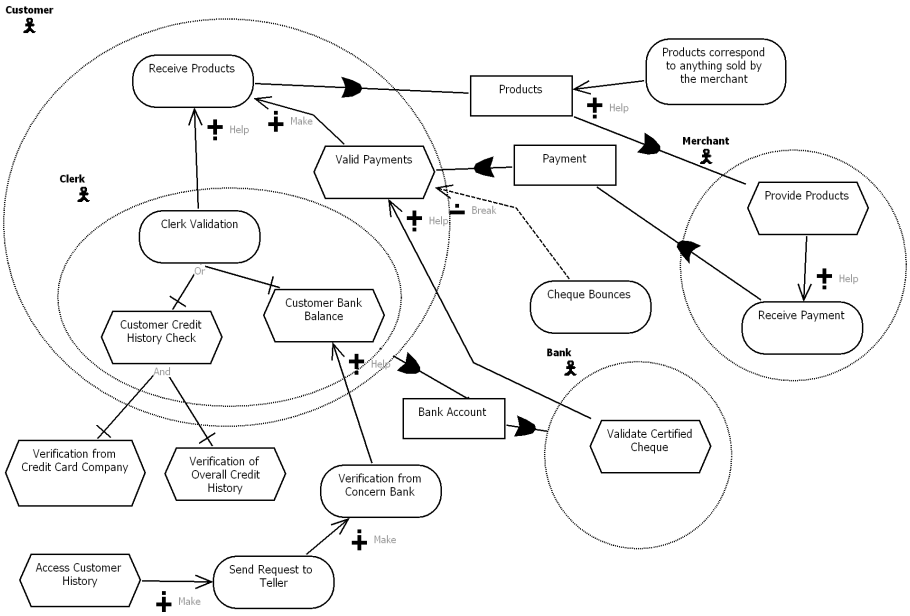

Fig. 2. GRL Model Example

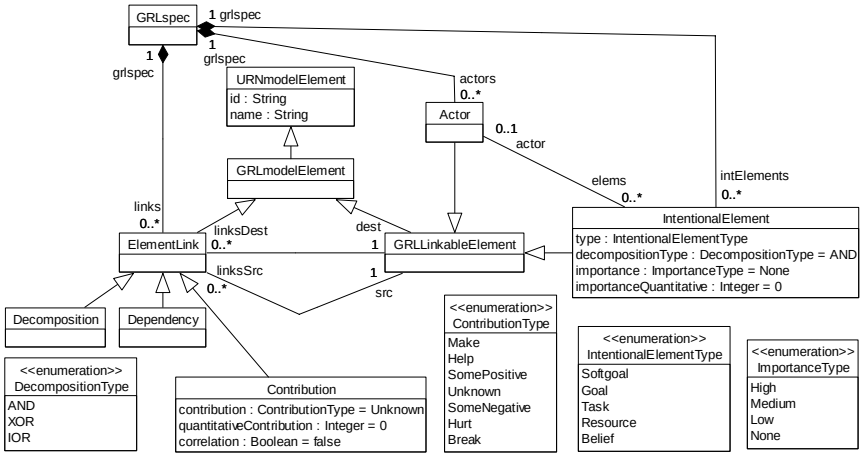

Fig. 3. GRL Metamodel (from Z.151)

ommendation Z.111 [9] for metamodel-based definitions of ITU-T languages. This recommendation standardizes notations used to create abstract and concrete syntaxes of languages, using metagrammars or metamodels. Z.111 uses a subset of the Meta-Object Facility [15] to define a simple meta-metamodel targeting the definition of modeling languages.

3 UML Profile for GRL

The UML metamodel is not intended to be directly modifiable outside of the UML standardizing process. However, *profiling* enables one to customize UML to a particular domain. A new domain-specific language can hence be defined and integrated to the core UML by creating a profile that extends the UML metamodel. There already exist different UML profiles that have been developed by the Object Management Group [17], including the Systems Modeling Language (SysML), the Enterprise Distributed Object Computing (EDOC), and the Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE). This section discusses the creation of a profile for goal-oriented modeling based on GRL's concepts.

3.1 UML Profile Creation

Two main approaches are used for creating UML profiles: the *Stereotype Mechanism* (SM) and the *Metamodel Extension Mechanism* (MEM). The Stereotype Mechanism is a very common and straightforward method for creating a UML profile. This approach extends the basic UML elements and is supported by most UML tools. The different constructs used to define a profile are *Stereotype*, *Tagged value* and *Constraint* [16]. A stereotype is a metamodel construct defined as an extension of an existing UML metamodel element, in a manner similar to class extension using inheritance. A tagged value is similar to an attribute construct of a class in a UML metamodel. Tagged values are standard meta-attributes. Finally, constraints are restrictions on a metamodel required in a particular domain. Notice that existing UML metamodel constraints cannot be weakened in a profile. The definition of a profile using the SM method consists of:

1. Assigning a new name to an extended metaclass, which will be represented as a *stereotype* of UML;
2. Adding new attributes in the stereotype, which are called *tags*;
3. Adding new *constraints* to the stereotype.
4. Assigning a new appearance to the stereotype.

One limitation of the SM approach is that it does not prevent the mixing of domain-specific diagram constructs with predefined UML diagram constructs. This can result in *diagram pollution* and potentially hurt the understandability of models.

The Metamodel Extension Mechanism [13], which includes the functionalities of SM, is a less common method of creating a UML profile. It allows the extension of non-basic UML elements and imposes restrictions that ensure the sole use of domain-specific stereotypes in diagrams, thus avoiding diagram pollution. While more flexible, the MEM is a more complex approach of profiling and is supported by fewer tools than the SM. Given the desirable properties defined in Sect. 1, MEM is favored over SM in our work.

3.2 Overview of UML Profile for GRL

Our profile is based on the conventions defined in the ITU-T Recommendation Z.119 [10]. This recommendation defines specific rules and a template for ITU-T languages profiling. Because of space restrictions, only a summary of the profile is provided in this paper. The complete profile, documented in accordance to recommendation Z.119, is available in Abid's thesis [1]. Table 1 provides a list of the stereotypes with the UML metaclass that each stereotype extends, while Figs. 4 to 7 graphically depict the UML profile with UML class diagrams.

Table 1. Summary of Stereotype Mapping Information

Stereotype	**Stereotyped UML Metaclass**
GRLspec	Model
GRLmodelElement	NamedElement
GRLLinkableElement	Class
Actor	Class
IntentionalElement	Class
IntentionalElementType	Enumeration
ImportanceType	Enumeration
ElementLink	Relationship
Contribution	Association
ContributionType	Enumeration
Dependency	Association
Decomposition	Association
DecompositionType	Enumeration

Metaclass *GRLspec* from Fig. 3 maps to a stereotype of the UML metaclass *Model* as shown in Fig. 4. This is consistent with the fact that *GRLspec* is intended to serve as a container for GRL specifications, a role fulfilled by instances of *Model* in a UML specification.

GRLModelElement, which serves as superclass for the elements of a GRL model, extends the UML metaclass *NamedElement* (Fig. 5).

The stereotype *GRLLinkableElement* is intended for generalizing GRL *Actor* and GRL *IntentionalElement*. *GRLLinkableElement* extends the UML metaclass

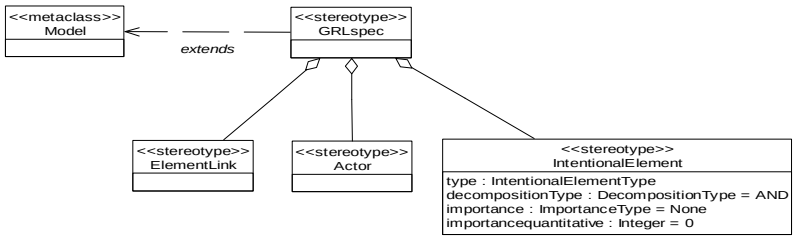

Fig. 4. GRL Specification

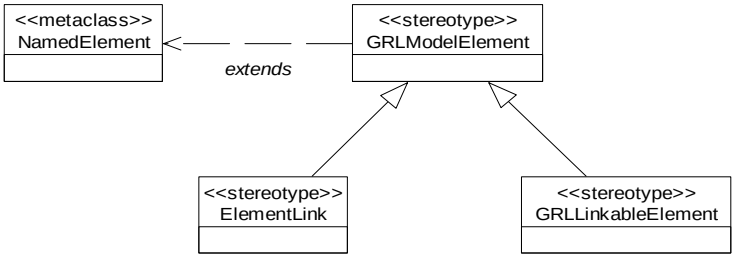

Fig. 5. GRL Model Element

Class (see Fig. 6). Stereotypes *Actor* and *IntentionalElement* are also defined as extensions to metaclass *Class*. The UML metaclass *Class* describes a set of objects that share the same specifications of features, constraints and semantics. Its features are comprised of attributes and operations. Similarly, a GRL *Actor* has attributes and operations. Association *elems* between *Actor* and *IntentionalElement* is captured as a subset of association *nestedClassifier* owned by *Class* toward *Classifier*.

The stereotype *ElementLink* extends the UML metaclass *Relationship* (see Fig. 7) as its purpose is to show the intentional relationship among *GRLLinkableElements* which include stereotypes *Actor* and *IntentionalElement*.

The specializations of *ElementLink* (*Decomposition*, *Dependency* and *Contribution*) are defined from the UML metaclass *Association*. *Contribution* includes additional tagged values for the contribution type, quantitative value and correlation.

The enumeration types used in the profile (DecompositionType, ContributionType, IntentionalElementType and ImportanceType in Fig. 3) are simply kept as is.

3.3 Sample Details for a Profile Metaclass: IntentionalElement

For each metaclass, Abid's thesis [1] provides subsections with the details of the corresponding attributes, constraints, semantics, notation, and references to UML, as recommended in Z.119. This section illustrates such details for the stereotype *IntentionalElement*.

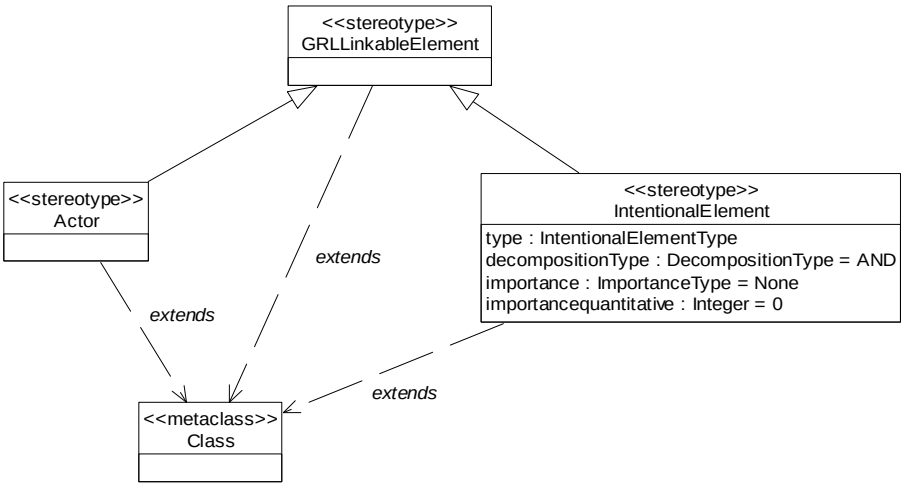

Fig. 6. GRL Linkable Element

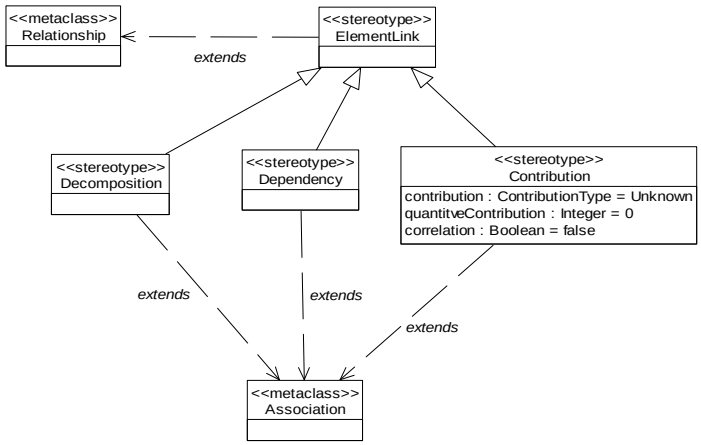

Fig. 7. Element Links

First, a correspondence between the attributes of the UML metamodel and those of GRL's is established. For example, *IntentionalElement* owns the attributes listed in Table 2 and defined as tagged values. This table also includes the attributes from inherited metaclasses (*URNmodelElement*, via *GRLmodelElement*, see Fig. 3).

Constraints on the stereotypes are then described in a separate subsection of [1]. For instance:

- «IntentionalElement» Class has a tag importanceQuantitative whose value must be ≥ 0 and ≤ 100.
- Each «IntentionalElement» Class must have a unique name.

The semantics is also informally described in a separate subsection of [1]:

- The «IntentionalElement» Class has an association with GRL «Actor». It specifies the reasons for including particular behaviors, information and structural aspects in a system's requirements. There are different types of intentional elements corresponding to different types of behavior and information elements. These various types have different notations and can be linked to each other.
- The «IntentionalElement» Class has a tag importance that captures an actor's level of interest in the included intentional element. However, it is not mandatory that modelers use both the importance and importanceQuantitative tags. The selection depends on a modeler's requirements for the desired analysis type, either qualitative, quantitative, or mixed.

The notation is also described for each metaclass of the profile. For example:

- An «IntentionalElement» Class has different types as mentioned above in semantics. Each type has a separate notation: [1]

In the last subsection of [1], references to the relevant sections of the UML superstructure [16] are included:

- UML Superstructure: 7.3.7 Class (from Kernel).
- UML Superstructure: 7.3.33 NamedElement (from Kernel, Dependencies).

4 Implementation and Validation

In order to validate our UML profile for GRL, we have implemented it using IBM Rational Tau version 4.0 [8]. This modeling environment supports UML profiling. It allows extending the UML metamodel and, therefore, enables modelers to customize the UML metamodel for specific domains. Both the stereotype mechanism (SM) and the metamodel extension mechanism (MEM) are supported by

[1] The five first symbols in Fig. 1.a would be described here, but we do not repeat them because of space constraints.

Table 2. Attributes of *IntentionalElement*

Attribute	**Description**
name: String	Defines the name of the «IntentionalElement» Class.
id: String	Defines the identifier of the «IntentionalElement» Class.
type: Intentional-ElementType	This is an *enumeration* data type. It defines the different types of GRL «IntentionalElement»: *Softgoal, Goal, Task, Resource* and *Belief*.
decompositionType: DecompositionType	This is an *enumeration* data type. Its possible values are *AND, XOR* and *IOR*. Its default or initial value is *AND*. It defines the *decomposition* type when GRL «IntentionalElement» is the source of the decomposition link.
importance: ImportanceType	This is an *enumeration* data type. Its possible values are *High, Medium, Low,* and *None*. Its default value is *None*. It is used to evaluate the importance level of the intentional element to its owning actor when specified.
importance-Quantitave: Integer	Defines the evaluation of the quantitative importance of GRL «IntentionalElement» on its GRL «Actor». Its value ranges from 0 to 100, with 0 as default.

the tool. However, some adaptation is required because the meta-metamodel supported by Tau (*TDDMetamodel*) differs from UML in a number of ways. For instance, the UML Enumeration and NamedElement metaclasses are absent or represented differently and the basic type String needed to be converted to Charstring. Hence, some customization of our profile was required during the implementation. Such differences in the meta-metamodel also leads to the generation of XMI files that may not be imported correctly by other UML tools, but tool interoperability was not a concern in our work so this was not tested.

Other UML tools supporting profiles were also considered: IBM Rational Software Architect was limited in its support of MEM [2], and Eclipse's Model Development Tools (MDT) was less mature than Tau at the time this work was done. However, MDT has evolved substantially since then, especially in terms of support for profiles and OCL-based validation of constraints [6].

Figure 8 gives an overview of the SM profile for GRL as defined in Tau. A drawback of the SM approach is that it does not allow the creation of a separate GRL editor. We must activate the GRL profile and then create a Class diagram. In the Class diagram editor, we create classes and manually associate an appropriate stereotype to each of these classes. A property view enables the setting of attribute values for the elements.

The MEM approach supported by Tau allows the generation of a specific GRL editor with a customized tool palette for GRL constructs. It is possible to "drag and drop" elements in the GRL editor to create a GRL model. The user is able to create a model based solely on GRL elements and avoid diagram pollution. Again, a property view can be used to provide values to attributes without visual representation. Furthermore, this approach allows to associate

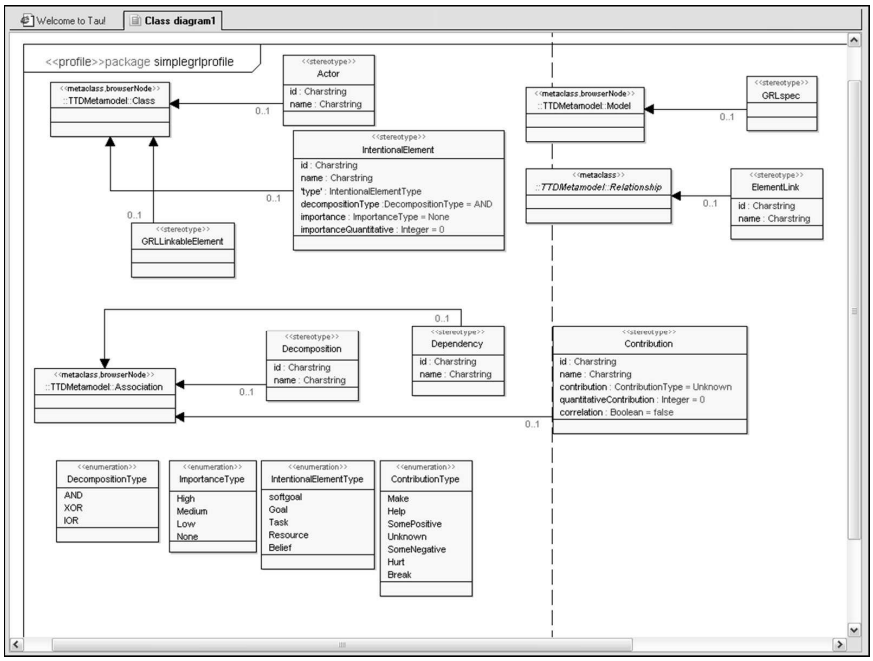

Fig. 8. UML Profile for GRL with SM in Tau 4.0

specific icons to the GRL elements. Figure 9 shows a view of a GRL editor created in Tau using the MEM approach.

Figure 10 gives a glimpse of a MEM-based implementation of the UML profile for GRL in Tau (which contains many more classes and connections to Tau's meta-metamodel than the SM metamodel). Such implementation is more complex as it requires one to connect the profile to predefined Tau stereotypes to allow/restrict visibility of elements in diagrams and the property view, and define icons, label positions, presentation, etc. In terms of semantics, this UML profile captures well the core GRL concepts described in Fig. 3, as these do not conflict with existing UML model elements. However, it is still incomplete as it does not yet support the notions of GRL strategy (which would be simple to add) and of connections to UCMs (on one hand harder to capture given their partial overlap with existing UML concepts, but on the other hand not absolutely required for goal modeling).

We evaluated our profile implementation by using the Tau generated editor on various examples, described in [1]. These experiments allowed us to verify the satisfaction of the properties listed in Sect. 1. In particular, GRL diagram elements are not intermixed with those from class diagrams (unlike the SM option). It is also possible to reuse GRL model elements in other diagrams: for example, one can drag a GRL actor from a GRL diagram to a UML use case diagram (to create an actor) or to a sequence diagram (to create a lifeline).

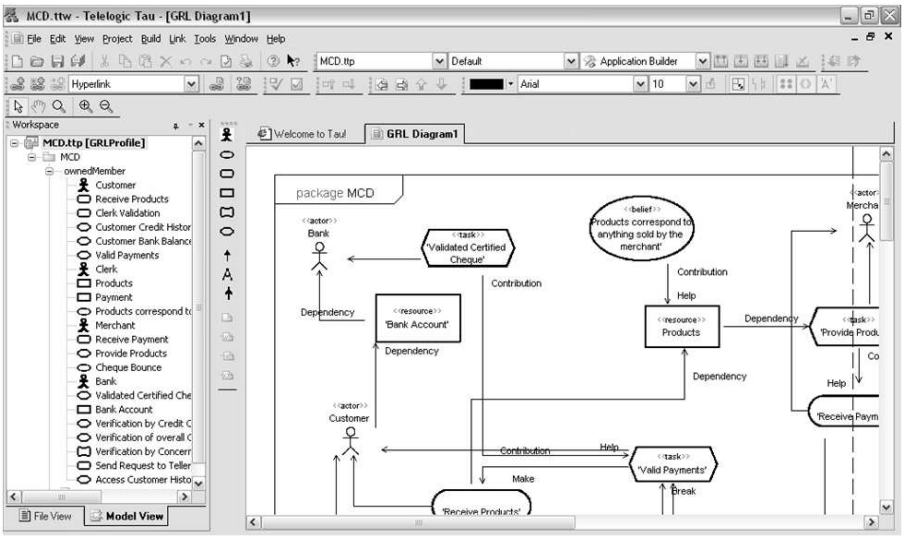

Fig. 9. Usage of UML Profile for GRL with MEM in Tau 4.0

Since these elements from different views refer to the same definition, changes to the name or another attribute of one of these elements are then automatically reflected in the other diagrams. Traceability links between GRL element and other UML elements can also be created and navigated. Such links are used to document design rationale and to explain decisions, so that weaker alternative designs are not constantly re-suggested. They also enable coverage assessments by answering questions such as "are all my goals sufficiently addressed by my UML design (goals without links)?" and "are there UML design elements that are spurious or that have no purpose (as they cannot be traced to any goal, directly or indirectly)?". This profile's implementation enables modelers to get answers to such important questions, which would be difficult to get otherwise.

We then compared the resulting diagrams with GRL diagrams obtained from jUCMNav [14], a dedicated URN modeling tool. A profile-based editor can hardly compete with a specialized editor from a usability perspective and in terms of how good the diagrams look (compare jUCMNav's GRL diagram from Fig. 2 with Tau's in Fig. 9). Yet, it is actually possible to capture goal concepts, attributes, and relationships from within a UML tool (with much of the syntax and a palette supported), and integrate them to the rest of the UML model.

We were also able to identify some limitations of Tau that had some impact on the appearance of the resulting diagrams (some of which were already reported in [2]). Among others, adding icons to qualify links (e.g., qualitative GRL contributions) and visually embedding GRL intentional elements inside the boundaries of actors proved to be impossible with the version of Tau we used (see Fig. 9). The constraints defined in the profile, discussed in Sect. 3.3, cannot be easily supported as there is no constraint language such as UML's

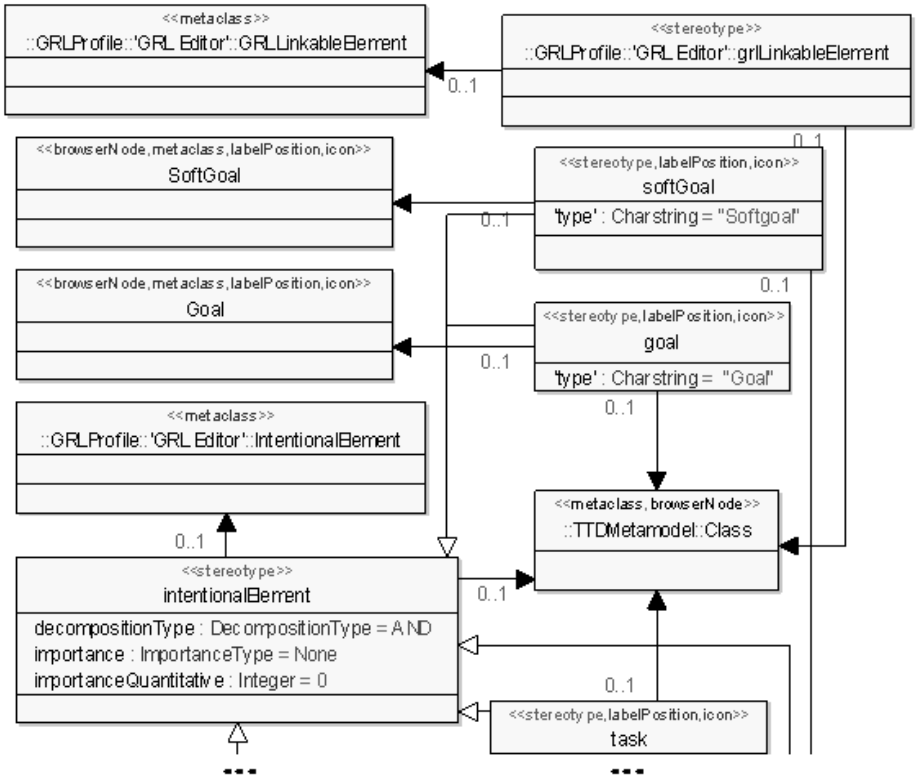

Fig. 10. UML Profile for GRL with MEM in Tau 4.0 (Extract)

OCL (although constraint verification procedures could be defined in a language like C++ using the notion of Tau *agents*).

5 Related Work

Little research has addressed UML profiling for goal modeling. Supakkul and Chung [18] proposed a metamodel for NFR concepts that is integrated with the UML metamodel through the use of a UML profile. The integration of UML and NFR notations occurs in a Use Case diagram. NFRs are represented as softgoals and associated with appropriate Use Case model elements. The authors implemented their work and illustrated it with a case study. A difference between this work and ours is that the former is restricted to NFRs. They also used an SM-based approach for the implementation, which does not prevent diagram (and concept) pollution.

A metamodel for enterprise goal modeling is proposed by Grangel *et al.* [7]. It distinguishes four different conceptual constructs: *Objective* (to represent targets that enterprises want to achieve), *Strategy* (to describe how the enterprise wants to achieve the proposed objectives), *Plan* (to represent the organization of the work at different hierarchal levels in order to accomplish the objectives and strategy) and *Variable* (to represent any factor that is able to influence the execution of the plans defined in the organization). A UML profile based on the metamodel has been implemented in IBM Rational Software Modeler Development Platform and in MagicDraw UML 12.0. Our goal metamodel appears more general and standard than the one in [7], which is specialized for enterprise goals. The profile implementation mechanism also appears to be an SM-based approach.

An approach for the integration of functional and non-functional requirements (NFR) is discussed by Cysneiros and Leite in [5]. The authors propose a junction point between the functional and the non-functional requirements development cycles named Language Extended Lexicon (LEL). The proposed approach is intended to cover all types of UML diagrams. For instance, a UML class diagram is integrated with related NFRs by having every root of each NFR graph refer to a LEL symbol and every class of the class diagram named using a LEL symbol. This approach is however not really defined as a UML profile. Similarly, van Lamsweerde presents an approach where goal modeling with KAOS is integrated with UML modeling, but not at the profile level [19].

6 Conclusion

In this paper, we presented a UML profile for goal-oriented modeling. Our research hypothesis is that UML can be profiled to support such modeling with a semantics rooted in a standard metamodel such as that of URN's Goal-oriented Requirement Language. A profile based on a mature and a stable metamodel that has been already used by editors and in analysis techniques is likely to represent a better alternative to the few existing solutions that currently exist and

that were reviewed in the previous section. The definition of such profile should also be done according to standard recommendations such as Z.119. Z.119 provided us with useful guidance on the structure and the level of detail required for a good profile, but it was limited in guiding the selection of suitable metaclasses in UML corresponding to our GRL concepts. The implementation of the profile also enables models that can combine goal-oriented concepts with object-oriented concepts in a way that is comprehensible by the UML community at large.

We demonstrated that UML can indeed be profiled in such a way, and that it is possible to provide corresponding tool support through a commercial, profile-enabled UML environment (Tau in our case). An implementation approach based on UML's metamodel extension mechanism further enables the integration of goal modeling concepts with UML's in a tool while avoiding diagram pollution. Overall, this satisfies the desirable properties specified in the introduction.

Future extensions of this profile should include support for GRL strategies, which describe what-if scenarios to analyze GRL models and which are part of URN. The profile should eventually be extended to encompass URN entirely by also including Use Case Map scenario concepts. This would be in line with ITU-T's objective of having UML profiles for its formal languages (one such profile exists for SDL, namely Recommendation Z.109). Improvements to UML tools supporting profiles would also help cope with the complexity of supporting such modeling notations. The scalability, usability and acceptance by modelers of such a profile are also important research questions that deserve deeper studies.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada, through its programs of Discovery Grants and Postgraduate Scholarships.

References

1. Abid, M.R.: UML Profile for Goal-oriented Modelling. Master of Computer Science Thesis, University of Ottawa, Canada (2008)
2. Amyot, D., Farah, H., Roy, J.-F.: Evaluation of Development Tools for Domain-Specific Modeling Languages. In: Gotzhein, R., Reed, R. (eds.) SAM 2006. LNCS, vol. 4320, pp. 183-197. Springer, Heidelberg (2006)
3. Amyot, D., Mussbacher, G.: Development of Telecommunications Standards and Services with the User Requirements Notation. Workshop on ITU System Design Languages 2008, http://www.itu.int/dms_pub/itu-t/oth/06/18/T06180000010012PDFE.pdf
4. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
5. Cysneiros, L.M., Leite, J.C.S.P.: Using UML to Reflect Non-Functional Requirements. In: CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research. IBM Press (2001)

6. Eclipse.org: Model Development Tools (MDT), http://www.eclipse.org/modeling/mdt/
7. 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 (2008)
8. IBM: Rational Tau, http://www-01.ibm.com/software/awdtools/tau/
9. International Telecommunications Union: Recommendation Z.111 (11/08), Notations to Define ITU-T Languages, http://www.itu.int/rec/T-REC-Z.111/en
10. International Telecommunications Union: Recommendation Z.119 (02/07), Guidelines for UML profile design, http://www.itu.int/rec/T-REC-Z.119/en
11. International Telecommunication Union: Recommendation Z.150 (02/03), User Requirements Notation (URN) – Language Requirements and Framework, http://www.itu.int/rec/T-REC-Z.150/en
12. International Telecommunication Union: Recommendation Z.151 (11/08), User Requirements Notation (URN) – Language definition, http://www.itu.int/rec/T-REC-Z.151/en
13. Jiang, Y., Shao, W., Zhang, L., Ma, Z., Meng, X., Ma, H.: On the Classification of UML's Meta Model Extension Mechanism. In: Baar, T., Strohmeier, A., Moreira, A.M.D., Mellor, S.J. (eds) UML 2004. LNCS, vol. 3273, pp. 54-68. Springer, Heidelberg (2004)
14. jUCMNav website, University of Ottawa, http://softwareengineering.ca/jucmnav
15. Object Management Group (OMG): Meta Object Facility (MOF) Core Specification, Vers. 2.0, http://www.omg.org/docs/formal/06-01-01.pdf
16. Object Management Group (OMG): UML Superstructure Specification, Vers. 2.1.2, http://www.omg.org/docs/formal/07-11-02.pdf
17. Object Management Group (OMG): Catalog of UML Profile Specifications, http://www.omg.org/technology/documents/profile_catalog.htm
18. Supakkul, S., Chung, L.: A UML Profile for Goal-Oriented and Use Case-Driven Representation of NFRs and FRs. In: Third ACIS International Conference on Software Engineering, Research, Management and Applications (SERA 2005), pp. 112-121. IEEE Computer Society (2005)
19. van Lamsweerde, A.: Requirements engineering: From System Goals to UML Models to Software Specifications. John Wiley & Sons (2009)
20. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE International Symposium on Requirements Engineering, pp. 226-235. IEEE Computer Society (1997)