A template-based analysis of GRL*

Gautier Dallons, Patrick Heymans, Isabelle Pollet

Computer Science Department, University of Namur
{gdl,phe,ipo}@info.fundp.ac.be

Abstract. The goal-oriented paradigm is widely popular in Requirements Engineering. However, the central notion of goal remains one of the most controversial in the field. A possible cause might be that research has devoted too little attention to studying the ontological foundations of goal-oriented languages. In this paper, we have studied the case of GRL, the goal-oriented requirements language being standardized by the ITU. Our analysis followed the template-based approach proposed by Opdahl and Henderson-Sellers. After defining a metamodel for GRL, we have applied the template to each of its constructs to extract and formalize detailed syntactic and semantic information. The semantic part of the template focuses on establishing a mapping between a construct and its *meaning*, defined in term of the Bunge-Wand-Weber ontology. Evaluations of both GRL and the template are provided as well as suggestions to improve them.

1 Introduction

In 2003, the UEML thematic network (IST-2001-34229, www.ueml.org) started to develop the so-called Unified Enterprise Modeling Language (UEML), a conceptual modeling language designed to be a common ground for (1) representing the various aspects of the enterprise and (2) facilitating the exchange of enterprise models. Reaching these objectives was deemed of utmost importance to improve the development, interoperability and integration of enterprise information systems. In 2004, UEML 1.0 was delivered. Due to the nature of the UEML project[1], UEML 1.0 allows mainly the modeling of process aspects but leaves out other aspects (such as static information, functional and non-functional requirements, resources and goals) and covers only a small set of the identified requirements. It was defined by integrating subsets of three existing Enterprise Modeling Languages (EMLs) – namely GRAI [3], EEML [6] and IEM [10] – by following a methodology inspired by database integration [13]. Development of the UEML has since then been taken over by the InterOP Network of Excellence [1]. It is currently an on-going activity carried out by a consortium made of

*This work is supported by the Commission of the European Communities under the sixth framework programme (InterOP Network of Excellence, Contract 508011) [1]

[1]The UEML project lasted only 15 months and its objectives were confined to demonstrate the feasibility of using UEML for exchanging models among three enterprise modeling software environments.

the leading practionners and researchers in the domain of Enterprise Modeling (EM). The adopted language development approach reconciles scientific rigour and pragmatism. First, it is a requirements document under continuous elaboration that drives the language development process. Second, commonly used EMLs are analysed, each in turn, according to a quality evaluation framework inspired by [8] in order to guarantee that not only the most used but also the most appropriate and sound constructs are incorporated in the UEML definition. Third, the integration of these constructs into the UEML is done in such a way that syntactic and semantic problems (widespread in other unified languages, such as UML) do not arise. Examples are synonymous, homonymous, underdefined, ill-defined, overly complex or poorly integrated constructs. Finally, evaluations of the successive versions of the language are performed to provide continuous feedback to the language development requirements and process.

The work reported in this paper describes some contributions of the authors towards the development of UEML 2.0. We focus on the analysis of existing EMLs and, more specifically, on the analysis of GRL, the goal-modeling language standardized by the ITU [4]. The language is presented in Sect. 2. The template-based approach [12] used to analyse EML constructs is sketched in Sect. 4. The analysis template is to be applied to every construct pertaining to a language's abstract syntax (aka metamodel). Since GRL does not have a proper metamodel, we had to define it ourselves (Sect. 3). The template-based analysis of GRL constructs follows (Sect. 5) and then undergoes discussion (Sect. 6). Because of space limitations, it was impossible to reproduce and discuss in this paper the analyses of all the GRL constructs that we performed but the companion technical report [2] provides the full details. We conclude with a summary and an outlook towards future work (Sect. 7).

2 GRL in a nutshell

GRL stands for Goal-oriented Requirements Language. It results from the integration of the *i** goal-modeling language [16] and the NFR framework [11]. The latter consists of a language and method designed to represent and reason about non-functional requirements (NFRs). Both languages originate from research performed at the University of Toronto and were among the first ones to consider goals as first-class citizens. However, each of them has a different focus. Indeed, NFR, which is a few years older than *i**, had for primary concern the modeling of NFRs and the various types of relationships between them (and- and or- decomposition, positive and negative contributions, etc.). NFR comes with goal decomposition strategies as well as propagation algorithms to estimate the satisfaction of higher-level goals given the (more measurable) attainment or non-attainment of lower-level ones. *i**, on the other hand, focuses on modeling the intentions of and strategic dependencies between actors. Dependencies between actors concern goals, softgoals, resources and tasks. It is the key concept of goal that makes the link between the two notations.

GRL is now in its third version and is a component of URN, the Unified Requirements Notation standardized by the ITU. URN has two main components : Use Case Maps (URN-UCM, see [5]) and GRL aka URN-NFR. Note that two other well-known and widely used languages are standardized by the ITU, namely MSCs and SDL.

Analysis of GRL was given a high-priority by the UEML development team (1) because the lack of goal and NFR modeling is currently a major drawback of UEML; (2) because of the popularity that the GRL notation is expected to gain through its development by an international standardization body such as the ITU; (3) because GRL is already the result of the integration of two complementary pioneering goal-oriented languages; and (4) because a relatively precise definition of GRL's syntax is public, which we have not found to be the case for other goal-oriented languages.

Details on the constructs of the language will be given in Sect. 2 where we report on their analysis. At this stage, we will just give an overview by quoting [4]:

> The URN-NFR language specified here is GRL [...], which is a language for supporting goal-oriented modeling and reasoning about requirements, especially non-functional requirements. It provides constructs for expressing various types of concepts that appear during the requirement process. There are four main categories of concepts: actors, intentional elements, non-intentional elements, and links. The intentional elements in GRL are goal, task, softgoal, resource and belief. They are intentional because they are used for models that allow answering questions such as why particular behaviours, informational and structural aspects were chosen to be included in the system requirements, what alternatives were considered, what criteria were used to deliberate among alternative options, and what the reasons were for choosing one alternative over the other. Actors are holders of intentions, they are the active entities in the environment or the system, who want goals to be achieved, tasks to be performed, resources to be available and softgoal to be satisficed. Links are used to connect isolated elements in the requirement model. Different types of link depict different intentional relationships. Non-intentional elements are equipped as a mechanism to refer to objects outside GRL model.

In addition to a general introduction from which the above paragraph is taken, the standard essentially has the following content:

- 3 concrete syntaxes for GRL : a textual syntax (expressed in BNF), a graphical syntax (expressed in BNF augmented with topological information) and an XML syntax (expressed with as an XML Document Type Definition (DTD));
- informal semantic definitions of constructs;
- examples of GRL models;
- a tutorial.

No abstract syntax is provided[2]. Thus, we provided our own that we elaborated from the sources listed above. Doing this, we encountered inconsistencies, ambiguities and underdefinitions. We will mention them as we go along and explain how we choosed to resolved them.

3 A metamodel for GRL

As a metamodeling language, we use standard UML Class Diagrams. Fig. 1 is the top-level view of the metamodel. The 4 main types of elements mentioned in the quoted paragraph in the previous section appear immediately: `Actor`, `IntentionalElement`, `NonIntentionalElements`, and link (that was renamed `IntentionalRelationship`, to be compliant with the syntax definition). Except `Model` and `ModelType`, which are generic to all the languages we plan to examinate, all the classes in this and subsequent diagrams are specific to GRL. We have located them in the GRL package in order to avoid confusion in a multilanguage context.

Fig. 2 details the 5 kinds of intentional elements: `Softgoal`, `Resource`, `Task`, `Goal` and `Belief`. Various abstract classes appear: `IEButBelief` (denoting all intentional elements except beliefs), `Correlator`, `Contributee` and `Contributor`. These latter classes are somewhat artificial. We introduced them for the sole purpose of showing graphically (vs. in OCL) and succintly the groups of classes which are likely to play a role wrt an intentional relationship (see Fig. 3). For example, in GRL, the contributee in a contribution relationship is either a belief, a softgoal or an intentional relationship. Therefore, we have introduced a superclass `Contributee` generalizing `Belief`, `Softgoal` and `IntentionalRelationship`. All such superclasses have stereotype `<<PossibleRole(s)>>` and are named after their corresponding roles (except `IEButBelief`, for brevity)[3].

The metamodel continues on Fig. 4. Details on selected constructs will be given in Sect. 4. For full details, please refer to the companion technical report [2].

4 Template-based analysis of modeling languages

In this section, we briefly present the template that we have used for analyzing the GRL constructs. The template was proposed by Opdahl and Henderson-Sellers [12] as a means to systematize the description of EML constructs. It can be used for various purposes like comparing and integrating EML constructs or, simply, for better understanding them. Translating models from one EML to another is another possible use.

In version 1.1 of the template (the latest at the time of writing), each construct is defined by filling in the following sections:

[2]However, the document indicates that this is foreseen.

[3]This avoids using less readable OCL constraints or overloading the diagram with a relationship for each subclass able to play a role

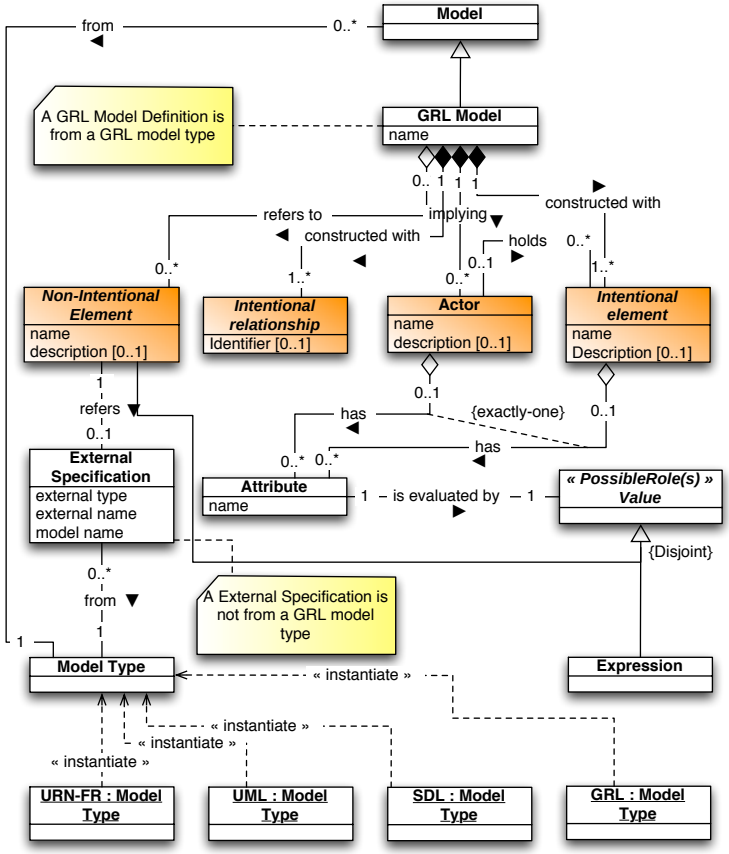

Fig. 1. Top-level view of the GRL metamodel

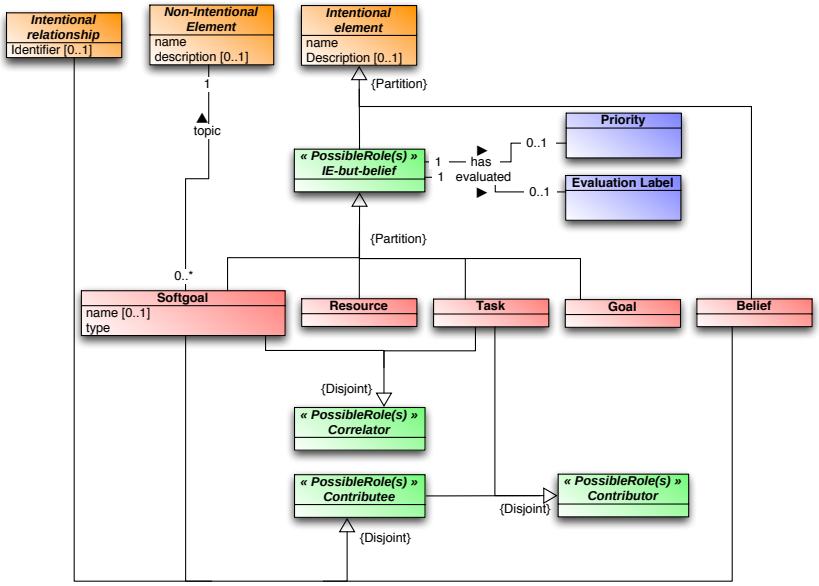

Fig. 2. GRL metamodel: zoom on intentional elements

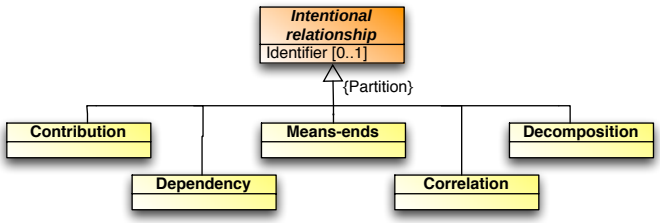

Fig. 3. GRL metamodel: types of intentional relationships

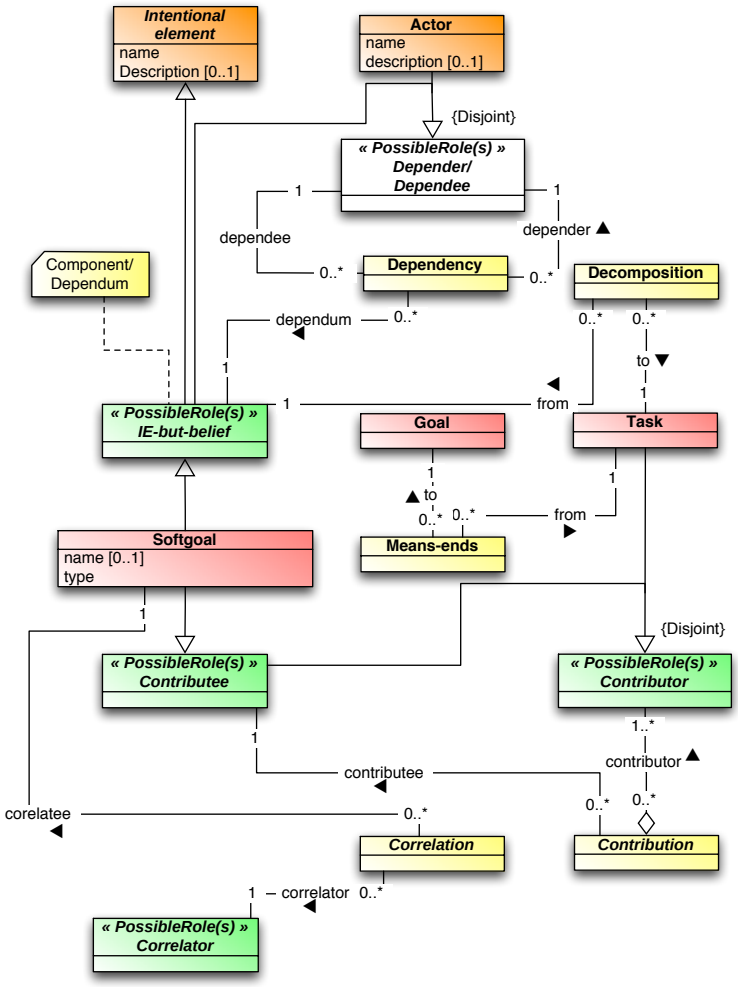

Fig. 4. GRL metamodel: zoom on intentional relationships

1. *Preamble.* General issues are specified here, i.e., construct, diagram type, language name and version, acronyms and external resources.
2. *Presentation.* Such issues as lexical information (icons, line styles), syntax and layout conventions are specified here.
3. *Semantics.* This section is the most important as well as the most complex. It requires the analyst to answer the four following questions:
   - Is the construct at the *instance* or *type* level?
   - Which *class(es) of things* in the world does the construct represent?
   - Which *property(-ies)* of those things does the construct represent? Things and properties are the basic building blocks upon which the other BWW concepts are built. Due to lack of space, we will only introduce such BWW concepts when we need them in the sequel.
   - Which *segment of the lifetimes* of those property(-ies) and things does the constructs represent? This question is only relevant for constrcuts denoting *behavioural* properties.
   - What is the *modality* of the assertions made using the construct? Is it that *something is the case* (regular assertion)? Is it that *somebody wants something to be the case*? Is it that *somebody knows that something is the case*? Etc.
4. *Open issues.* All the issues that the template failed to address should be mentioned here.

The section devoted to semantics is based on the Bunge-Wand-Weber (BWW) ontology [15], a now widespread reference for the semantic evaluation of information system concepts. However, this ontology, despite recent efforts to formalize it and make it more accessible [14], remains complex, sometimes ambiguous and not so well-known by EM experts. One of the main advantages of the template is that is does not require its users to be BWW nor ontology experts. It helps relate EML constructs to the the abstract categories of BWW by asking simple questions, giving practical recommendations and providing concrete examples.

5 Template-based analysis of GRL constructs

Due to space limitations, we cannot describe in detail the analysis of each construct. The interested reader will find the detailed analyses of the other constructs in the technical report [2]. In this section, we skip the trivial syntax-related points to concentrate on the semantics (section 3 of the template). Our analysis is summarized on table 1. The leftmost column lists the GRL constructs. The upper part of the table lists the intentional elements while the lower part is devoted to intentional relationships. If a construct is mapped to a BWW class, this is indicated in the middle column. If it is mapped to a BWW property, this is indicated in the rightmost column. Note that we do not claim that this semantic mapping is better than any other. We have tried to be as faithful as we could to the GRL and BWW definitions but, in the end, it remains the product of our subjectivity. It is exposed here for the purpose of being discussed with peers.

The `Actor` construct is mapped to the BWW *ThingsActingOnTheProposedSystem* class. In BWW, "acting" has a broad meaning. This mapping entails that an actor is someone or something (see Sect. 6) having some influence on the future system. Similary, the `Resource` construct is mapped to the *ActedOnThings* class. In our mapping, this class means that a `Resource` is a thing on which someone or something can act.

The `Goal`, `Softgoal` and `Belief` constructs are all mapped to a *state law property*. For `Goal` and `Softgoal`, this is explained by the fact that these constructs were deemed to constrain the possible states of the proposed system: an actor wants something to be true in the proposed system, thereby constraining the possible states of the system. As an example, if we have a goal "increase the number of orders taken in charge" then there is a constraint on the possible future system states. A `Belief` is also a state law but true in both the current and future world (with or without the proposed system). `Goal` and `Softgoal` have been given a modality representing that their corresponding state law are *wanted* by some actor. `Belief`, like all other GRL constructs, is just a regular assertion.

`Task` is mapped to a *transformation law*. Indeed, a task will have an impact on the system and will hopefully result on a change of the state of world.

The `Means-End` relationship is also mapped to a *transformation law*. The end is an objective to be achieved by the proposed system and the means is the way of achieving it. So, `Means-End` defines a transformation from the current system/state to a next state closer to the future system. `Decomposition` is also a *transformation law*. For example, a system with only one task evolves towards a system with several tasks which are the sub-task of the previous one. We understand `Means-End` and `Decomposition` as two kinds of system refinements.

The `Dependency` relationship is mapped to a *state law*. This relationship denotes the dependency of an actor on another wrt an object of dependency, called the dependum. The state of the dependum is constrained by the dependency between actors. Hence a state law.

Finally, `Contribution` and `Correlation` are mapped to *mutual property*. A contribution is a shared property between two coupled objects. In BWW terms, this amounts to a mutual property. A correlation is similar, except that it does not happen by design but by side-effect.

6 Discussion

6.1 Assessment of GRL

First, we recall the subjectivity of the results exposed in the previous section. This is actually reinforced by the fact that the GRL specification is quite imprecise. Indeed, in the specification, we found only very broad semantic definitions and the tutorial does not help us to precise them. Most of the time, our interpretation has played a key-role in the understanding of GRL constructs. This could be seen as a force since, this way, the GRL application domain remains

Table 1. Summary of the GRL template analysis

GRL construct	Class entry	Property entry
`Actor`	*ThingsActingOnTheProposedSystem*	-
`Goal`	-	*State Law*
`Task`	-	*Transformation Law*
`Softgoal`	-	*State Law*
`Resource`	*ActedOnThings*	-
`Belief`	-	*State Law*
`Means-ends`	-	*Transformation Law*
`Dependency`	-	*State Law*
`Decomposition`	-	*Transformation Law*
`Contribution`	-	*Mutual*
`Correlation`	-	*Mutual*

vast. From our point of view, this is a weakness because we were left with many questions which could be a major impediment if one has to make a concrete GRL model or to transform an existing GRL model into another notation. For example, in [4], an `Actor` is defined as *an active entity that carries out actions to achieve goals by exercising its know-how.* What about a group of people? Is it an actor? Can we consider types of actors? Are actors always people or can they be things (like expert systems)? Are roles also considered actors as in *i**. Having no answer, we stayed with the broadest interpretation but miss a way to differentiate these concepts in the language. Similar questions were raised for the other constructs as well. They can be found in the technical report.

Another problem we encountered is the existence of contradictions between the concrete syntaxes from which we had to build the metamodel. We had to make choices that do not necessarily represent the intentions of the GRL authors. For example, the textual syntax sometimes allows so-called short-hand forms which compliance with the informal semantics was deemed doubtful. An example is decomposition. In the text, only tasks are said to be decomposable. However the syntax allows a shortcut where a goal can be decomposed. In this case, we have decided to stick to the text and ignore the syntax definition. The meta-model presented in this article was constructed in this spirit.

Finally, we think the textual syntax could be improved especially wrt to the chosen keywords which are not always intuitive. For example, the syntax of decomposition is defined by the following rule: ***DECOMPOSITION*** *Optional Identifier* ***FROM*** *sub-element* ***TO*** *Decomposed Element*
We think the **FROM** and **TO** keywords are quite misleading in this order. A more intuitive definition could be: ***DECOMPOSITION*** *Optional Identifier* ***OF*** *Decomposed Element* ***INTO*** *sub-element*

6.2 Assessment of the template-based approach

For our purpose, the template was found to be very useful. It helped to raise a number of important issues about the analysed language which have been exposed in the previous sections. We particularly appreciated its ease of use even for those who, like us, are not BWW experts. Some familiarity was quickly gained by browsing through the many available examples, especially the analyses of well-known UML constructs. Still, we think that some improvements could be brought to the template.

First, it appeared that the predefined BWW concepts were quite broad. They were more or less sufficient because the semantic description of GRL is itself quite vague and wide-embracing. However, we encountered some problems. For example, a goal and a dependency are both BWW state laws although they are very different concepts. Hence, it is important to still refer to the initial less formal definitions to understand the differences. It would become quite dangerous to compare directly two concepts mapped to a single BWW class without looking at the original definitions, unless new BWW classes can be defined by the user. This was not attempted in this first use of the template. Similarly, although we did not detail the various contribution and correlation subtypes at this stage, we foresee the same problem to occur here.

Another point is the modality field. Currently, it only asks whether the assertion is modal or not. It was sufficient because GRL is not very precise here either but a more fine-grained list of modalities could be provided. This could be useful, for example, if we had to capture categories of goals such as those proposed by [7] or [9]. More details are given in the Related Work section of the technical report.

Finally, we think that tool support could be of great help to fill in the various entries. It could give more guidance (by restricting the possible values), allow safer reuse than copy and paste (which was heavily used throughout the analysis and was the source of many mistakes), and directly create the links between BWW and metamodel elements (which would facilitate other automated treatments and visualisations).

7 Summary and future work

In this paper, we have reported on the experimental analysis of the GRL language through the template-based approach defined by Opdahl and Henderson-Sellers. Despite its simplicity and its discussed limitations, the template allowed us to identify a number of pending important issues in the current GRL specification. We have also proposed a metamodel for GRL which was not available beforehand. Modulo some improvements, we think that the template is likely to scale up to be a solid basis for the analysis and comparison of Enterprise Modeling Languages needed for the elaboration of UEML 2.0. However, due to the amount of subjectivity that we had to put into the analysis, we first need to discuss our results with peers before we can reach a stable consensus.

In the future, we plan to improve the analysis with the feedback obtained and go deeper into the exploration of constructs that necessitate the creation of custom BWW definitions. Other Enterprise Modeling Languages will be analysed. Then, we will proceed to the selective integration of the analysed languages and constructs into UEML 2.0. For this larger scale applications, tool support is deemed crucial and will be investigated readily.

Acknowledgements: We thank Andreas Opdahl for the time he spent sharing his knowledge and reviewing our analysis.

References

1. INTEROP project website. http://www.interop-noe.org/, April 2004.
2. Gautier Dallons, Patrick Heymans, and Isabelle Pollet. A template-based analysis of grl. Technical report, University of Namur, March 2005.
3. G. Doumeingts. *GRAI: Méthode de Conception Des Systèmes En Productique*. PhD thesis, University of Bordeaux I, France, 1984. in french.
4. ITU. Recommendation z.151 (grl) - version 3.0, September 2003.
5. ITU. Recommendation z.152 (ucm) - version 3.0, September 2003.
6. H.D. Jorgensen and S. Carlsen. Emergent workflow: Integrated planning and performance of process instances. In *Proc. Of Workflow Management'99. Münster, Germany*, 1999.
7. Evangelia Kavakli. Goal oriented requirements engineering: A unifying framework. *Requirements Engineering Journal*, 6(4):237–251, 2002.
8. J. Krogstie and A.Sølvberg. Information systems engineering: Conceptual modeling in a quality perspective. Technical report, NTNU, January 2 2000.
9. Emmanuel Letier. *Reasoning about Agents in Goal-Oriented Requirements Engineering*. PhD thesis, Universit Catholique de Louvain, 2001.
10. Kai Mertins and Roland Jochem. *Quality-Oriented Design of Business Processes*. Kluwer Academic Publishers, Boston/Dordrecht/London, 1999. ISBN 0-7923-8484-9.
11. John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Trans. Softw. Eng.*, 18(6):488–497, 1992.
12. Andreas L. Opdahl and Brian Henderson-Sellers. A template for defining enterprise modeling constructs. *J. Database Manag.*, 15(2):39–73, 2004.
13. Michaël Petit. Some methodological clues for defining a unified enterprise modelling language. In Kurt Kosanke, Roland Jochem, James G. Nell, and Angel Ortiz Bas, editors, *Enterprise Inter- and Intra-Organisational Intergration - Building an International Consensus*. Kluwer Academic Publishers, kurt kosanke, roland jochem, james g. nell and angel ortiz bas (editors) edition, 2003. ISBN 1-4020-7277-5.
14. Michael Rosemann and Peter Green. Developing a meta model for the bunge—wand—weber ontological constructs. *Inf. Syst.*, 27(2):75–91, 2002.
15. Yair Wand and Ron Weber. An ontological model of an information system. *IEEE Trans. Softw. Eng.*, 16(11):1282–1292, 1990.
16. Eric S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, page 226. IEEE Computer Society, 1997.