*i** on ADOxx®: A Case Study

Margit Schwab[1], Dimitris Karagiannis[1], Alexander Bergmayr[1]

[1] Universitaet Wien, Faculty of Computer Science,
Department of Knowlede and Business Engineering,
Bruennerstrasse 72, 1210 Vienna
{ms, dk, ab}@dke.univie.ac.at

Abstract. Based on the ADOxx® meta-modelling platform, the conceptualization of the *i** method is discussed by means of a case study. The focus lays on the "translation" of *i** concepts into a conceptual model leveraging the instantiation of meta-classes provided by the utilised ADOxx® platform. Thereby the consideration of all concinnities of both the *i** meta-model and corresponding instance models within a specific domain is essential. The claim is that the ADOxx® platform supports this with adequate abstraction mechanisms. When using a meta-modelling platform, the first step of semantic integration can be achieved, where the modelling language and the platform use a meta-modelling approach as a concept. The result of this case study is accessible on www.openmodel.at/istar.

Keywords: meta-model, modelling language, *i**, ADOxx®, meta-modelling platform, method-engineering and -engineer;

1 Introduction

The construction of models and by their means processing particular information is nowadays a common procedure. Depending on the domain, the purpose and the underlying meta-model, the resulting instance models are "by definition" more or less complex. We use the classification of model hierarchies and language levels according to Kühn [11, p 32]. As we speak of instance models we think of graphical models and refer to the distinction of different types of models [9, 10, 8]. The end user of the meta-model, let's call him/her modeller, will use the modelling method at hand ideally in terms of the method engineer. In addition s/he will shape and refine the information to be conveyed with the instance model in the best possible way. This task is in general at least twofold.. Firstly, there is the design of the model and secondly, there is nearly at all times the need to consider further concinnities, like additional model descriptions in natural language, time-related data, necessary skills during processing, or applicable forms or regulations of the model. In fact the effort spent for the latter varies depending on the purpose and the target group the instance model is designed for. This demands flexibility from the underlying meta-modelling platform, in concrete for the case study ADOxx®. Although there are already a number of solutions available providing an implementation of this method [2], the crucial distinction feature in this study case is that the design and realization of the *i** method [6, 16, 19] is based on a meta-modelling approach, as described in Section 2. Section 3 is devoted to lessons learned. Section 4 concludes the paper and gives an outlook on further research questions to be addressed.

2 The *i** Method Case Study

In this case study the *i** method on one hand and the ADOxx® meta-modelling platform on the other hand have been used. The former provides a specification for the syntax, the semantic and the notation. The method concepts: *actor, role, agent* and *position* are subsumed under the term "intentional actor". Furthermore there are the elements *goal, softgoal, task, resource* and *belief.* These elements form the group "intentional elements". Connections comprise the constructs of a *dependency link*, *association link*, *means-end link*, *decomposition link* and *contribution/correlation link* [6, 19]. On the other side the ADOxx® supports the process of method customization and allows the - mostly graphical - creation, persistence, maintenance and usage of models. Further it offers functionality to freely define and configure arbitrary meta-models of modelling languages including the definition or adaptation of the corresponding procedures and mechanisms applicable to models. By means of these tools the method engineer elaborates the translation of the *i** method into a conceptual model. The result of this development is a semi-formalised structure of the available modelling concepts and their dependencies. For the construction of graphical models the *i** method also provides integrated mechanisms, especially to perform goal satisfaction evaluations, based on these models. For these mechanisms further requirements related in particular to the notation of the modelling concepts can be specified during the developing phase by the method engineer. In the following the focus lays on the elaboration of the *i** conceptual model, in the translation part and the customization concepts, in the instantiation part [1].

2.1 Part I: The Translation

The starting point for the translation is the ADOxx® **metameta-model** which exhibits the structure for the matching of *i** concepts. In the following selected concepts of this metameta-model are used to exemplarily demonstrate mappings between the method and the platform. The method engineer has to know these concepts in order to fulfil the intellectual process: **to design a holistic view of the *i** conceptual model** (see Fig.1.).

The first applicable concept is **library.** The library is a container to which all formalisms and constructs of an instance of a modelling language are assigned to. Yet, the meta-model possesses also a particular structure so that the assigned elements are not loosely arranged abreast on an equal level. The next step is to allocate the constructs of the modelling language to **model types.** The *i** method comprises two different types of models, the *strategic dependency model* and the *strategic rationale model*. The model type is a modularisation element for the available modelling concepts of a method. Hence, a model type strategic dependency model groups for example all modelling concepts necessary to map strategic dependencies for a particular scenario.

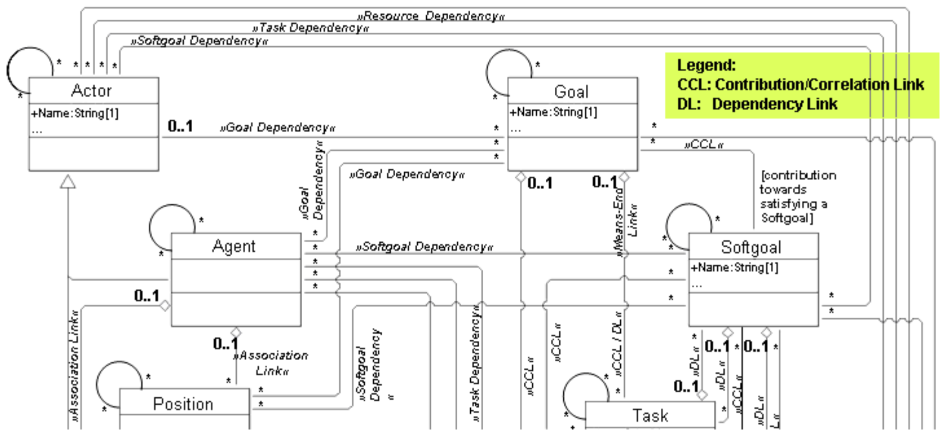
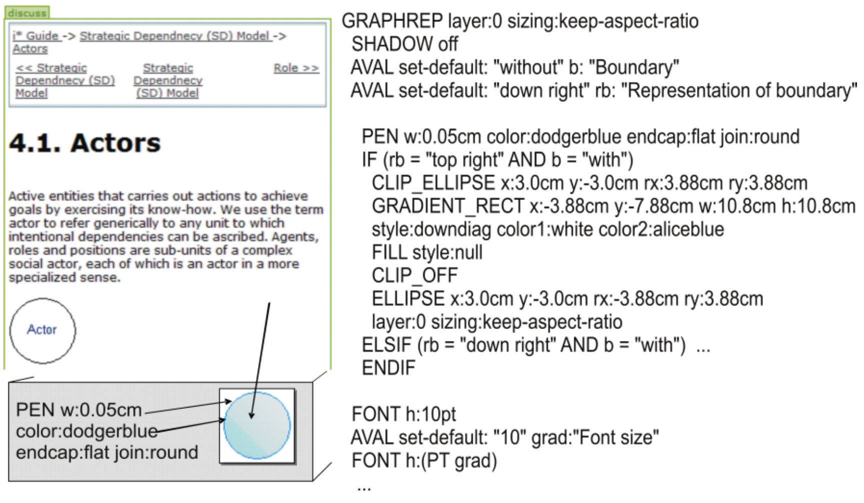


Fig. 1. Holistic View of the *i** Conceptual Model [part]

The modelling concepts are described with **classes** which are assigned to a particular model type. In ADOxx$^{®}$ **modelling classes**, which are in the *i** method, *actor, agent, role etc.* and for the **relation classes**, which are *dependency link, association link etc.* are distinguished.The different classes have particular properties. In this point the *i** method gives the method engineer an opportunity of a precise formal description about the syntax of the classes by means of **attributes.** The only mandatory requirement of the platform is that each class, modelling class or relation class, has a **name attribute**, because technically speaking, it becomes a global identifier. We distinguish between class attributes and instance attributes. The differene between these two lay in the values the attribute can adopt. Class attributes are context neutral and not to be filled by the end user or modeller using the method after implementation. Instance attributes are context dependent and will be used by the modeller to capture data and convey certain information [11, p. 100]. The **attribute type** is determined by the value the attribute can adopt when using the method, i.e. which data should be captured. Beside commonly known datatpyes ADOxx$^{®}$ additionally provides support for inter model references, expressions, tables, or programm calls to name some of them. This list can be extended for applying the algorithms and mechanisms on the instance models.

2.2 Part II: The Instantiation

The instantiation should lead to a mapping between the ADOxx$^{®}$ meta-classes with the *i** modelling and relation classes. After this step the customizing effort can be determined. The customizing is conducted on the level of the modelling language - considering the notation, the syntax and the semantic - on the method procedure level as well as from the processing point of view within mechanisms and algorithms. For demonstrating the work which is involved is shown in an example concerning the customizing of the notation.


discuss

i* Guide -> Strategic Dependnecy (SD) Model -> Actors

<< Strategic Dependnecy (SD) Model   Strategic Dependnecy (SD) Model   Role >>

4.1. Actors

Active entities that carries out actions to achieve goals by exercising its know-how. We use the term actor to refer generically to any unit to which intentional dependencies can be ascribed. Agents, roles and positions are sub-units of a complex social actor, each of which is an actor in a more specialized sense.

```
PEN w:0.05cm
color:dodgerblue
endcap:flat join:round
```

```
GRAPHREP layer:0 sizing:keep-aspect-ratio
 SHADOW off
 AVAL set-default: "without" b: "Boundary"
 AVAL set-default: "down right" rb: "Representation of boundary"

  PEN w:0.05cm color:dodgerblue endcap:flat join:round
  IF (rb = "top right" AND b = "with")
    CLIP_ELLIPSE x:3.0cm y:-3.0cm rx:3.88cm ry:3.88cm
    GRADIENT_RECT x:-3.88cm y:-7.88cm w:10.8cm h:10.8cm
    style:downdiag color1:white color2:aliceblue
    FILL style:null
    CLIP_OFF
    ELLIPSE x:3.0cm y:-3.0cm rx:-3.88cm ry:3.88cm
    layer:0 sizing:keep-aspect-ratio
  ELSIF (rb = "down right" AND b = "with")  ...
  ENDIF

 FONT h:10pt
 AVAL set-default: "10" grad:"Font size"
 FONT h:(PT grad)
  ...
```

Fig. 2. Actor *i** Notation: Graphrep Representation

The *i** method gives guidelines how the different classes look. It specifies the shape. Furthermore that for the modelling classes the value of the name attribute is displayed in the centre of the shape. The intentional actors can furthermore possess a *boundary*. The intentional actors are represented with a boundary, to express that all intentional elements within this boundary are explicitly desired by that intentional actor. For the relation classes association link and contribution/correlation link several type of links, for example if the association link is a "covers", "plays" etc. association, are specified. They are visualised with a respective label. This requirement can be fulfilled with the **Graphrep** formalisms which are provided by ADOxx®. Fig.2. gives an example of the actor specification and the realization in the ADOxx® Graphrep. In analogy, the customization effort for syntactical and semantical requirements is fulfilled through ADOxx® functionality. The version of the *i** method which has been translated and customized on the ADOxx® platform as described in the case study at hand is available on the open models platform.

3 Some Lessons Learned

The *i** method provides a high degree of maturity in terms of method specification. Nevertheless, the mapping on the meta-modelling platform ADOxx® showed that the offered platform functionality gives new input to further conceivable extensions. Suggestions for extensions rely on the analysis of instance models provided in different papers and can be structured by *extensibility* for syntax and notation as well as by *interpretability* [16, 17].

Extensibility. The syntax is related to the notation as some attributes are only necessary for the "orchestration" of a certain graphical representation, for example that the actor is represented with a boundary. The actor boundary belongs to a very specific actor and if a boundary is required to express the delineated semantic, it should be possible that the modeller can activate and deactivate the boundary on the drawing area as needed. In ADOxx® this can be done by defining an attribute, e.g. boundary and two possible predetermined values 'with' or 'without'. Another extension concerns the colour as it is an important distinction element, to keep the available shapes and as a consequence their meaning apart, there is the suggestion to use coloured elements. The reason for this is that if the instance models are of a certain size they tend to become hard to overlook and to read. From the experience of working with end users and addressees of instance models we know that the rectilinear an instance model is mapped, the easier the reader picks up the content and the better s/he understands the scenario captured with it.

Interpretability. During the analysis phase of instance models it became obvious, that the two previously identified model types – strategic dependency model and strategic rational model – are candidates for applying the ADOxx® view concept. As both types use most of the available classes of the *i** method mutually. The latter refines the former entirely or only in parts by using the same instance objects. Hence, what in the *i** method is expressed by two different types of models is considered as a view or **mode** in the meta-model of the ADOxx® platform. Despite of using the mode concept at least one model type needs to be defined. The suggestion for a name of the newly specified model type is ***Intentional actors and elements model.*** As the name should be specific for the *i** method and should convey as precisely as possible the context that should be documented with a strategic dependency model and a strategic rational model. The new model type has two modes, once the strategic dependency mode and twice the strategic rational mode whereas the former is defined as the default mode.

The explained lessons learned are concerning the conceptualization of a particular method. The deployment of the instantiated method is beyond the focus of the case study, although there are related technical questions the method engineer needs to answer in order to provide a "ready-to-use" tool. The ADOxx® platform offers respective support for this task.

4 Conclusions

Instance models and with them the modelling method they have been created with are commodity, this is also valid for the *i** method. There will always be the need for new methods or extended functionality of existing ones in order to convey information in the intended way. Once modelling methods are to be used by a broader group of people and drift off the initial inventing team it becomes advantageous if the method is supported by a tool. The more the modelling method avoids ambiguity in expressing certain information the more difficult it is to "translate" this modelling method and support it by a platform. From our experience one reason for that is that today's meta-modelling platforms lack in providing the full range of required functionality. Further research on the side of meta-modelling platforms and furthermore on the end user side is essential. This even more if this is seen in context of the integration of different modelling methods. For the later this is a claim in form of a non-functional requirement with regards to the utilisation of the modelling method by the end user resulting in user-friendly handling and as a consequence user acceptance [5, 18].

Acknowledgments. The authors thank all the colleagues from the University of Vienna and the Open Models Community for helpful discussions and comments during the realization of this case study.

References

1. Karagiannis, D.; Kühn, H.: "Metamodelling Platforms"; in Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2 – 6, 2002, LNCS 2455, Springer Verlag, Berlin Heidelberg, p 182 ff.
2. List of *i** tools, http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Tools, last access 5th of March 2010.
3. Cares, C., Franch, X., Perini, A., Susi, A.: „iStarML The i* Mark-up Language: Reference's Guide; Barcelona, Spain, August 2007.
4. Franch, X., Grau, G.: "Towards a Catalogue of Patterns for Defining Metrics over i* Models", CAiSE 2008, Springer-Verlag Berlin Heidelberg 2008, LNCS 5074, pp. 197–212.
5. Mylopoulos, J., Chung, L., Yu, E.: "From Object-Oriented to Goal-Oriented Requirements Analysis", Communications of the ACM, Vol. 42, No. 1, January 1999.
6. Yu, E.: "Strategic Actor Relationships Modelling with i*, Part 1, Part 2, Part3, A tutorial given at IRST / University of Trento, Italy, December 2001; http://www.cs.toronto.edu/~eric/#istar-tut-ppt; last access 25th of February 2010.
7. Open Models Initiative, http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf; last access 25th of February 2010.
8. Open Models Initiative; http://www.openmodels.at/web/istar/1-5; last access 25th of February 2010.
9. Harel, D., Rumpe, B.: "Meaningful Modeling: What's the Semantics of 'Semantics'?". In: IEEE Computer, Vol. 37 No. 10, 2004, 64-72.
10. Strahringer, S.: Metamodellierung als Instrument des Methodenvergleichs. Shaker, Aachen, 1996.
11. Kühn, H.: "Methodenintegration im Business Engineering"; Dissertation, Universität Wien, April 2004.
12. Braun, C., Wortmann, F., Hafner, M., Winter, R.: "Method Construction – A Core Approach to Organizational Engineering", in Applied Computing 2005, Proc. of the 2005 ACM Symposion on Applied Computing, Santa Fe, New Mexico, USA, 13.03.2005, ACM Press, New York, NY, USA, 2, 2005, pp. 1295-1299.
13. Tan Jun: "Software Develop Method Construction A Kernel Approach to Organizational Engineering". In Software Engineering WCSE '09, WRI World Congress on Software Engineering, Volume 4, May 2009.
14. Becker, J.; Holten, R.; Knackstedt, R.; Neumann, S.: Konstruktion von Methodiken – Vorschläge für eine begriffliche Grundlegung und domänenspezifische Anwendungsbeispiele, Institut für Wirtschaftsinformatik, Universität Münster, 2001.
15. Software Engineering Institute, SEI, "Introduction to the Architecture of the CMMI® Framework", http://www.sei.cmu.edu/reports/07tn009.pdf, last access 5th of March 2010.
16. Fazel-Zarandi, M., Yu, E.: "Ontology-Based Expertise Finding", In Proceedings of the 7th International Conference of Practical Aspects of Knowledge Management, Yokohama, Japan, 2008. Springer-Verlag, Berlin Heidelberg (2008), pp. 232-243
17. Samavi, R., Yu, E. Topaloglou, Th.: "Strategic Reasoning about Business Models: A Conceptual Modeling Approach", Information Systems and E-Business Management, Springer , Berlin / Heidelberg, Vol. 7, No. 2, March 2009.
18. Mylopoulos, J., Chung L., and Nixon, B., "Representing and using Non-functional Requirements: A Process-oriented Approach", IEEE Transactions on Software Engineering, June 1992.
19. Yu, E. S.: "Social Modeling and i*", in "Conceptual Modeling: Foundations and Applications", Borgida, A. T., Chaudhri, V. K., Giorgini, P., Yu, E. S. (Eds.): Mylopoulos Festschrift, LNCS 5600, Springer Verlag, Berlin Heidelberg, June 2009, p 99.