VIDEAS: A Development Tool for Answer-Set Programs based on Model-Driven Engineering Technology*

Johannes Oetsch[1], Jörg Pührer[1], Martina Seidl[2,3], Hans Tompits[1], and Patrick Zwickl[4]

[1] Technische Universität Wien, Institut für Informationssysteme 184/3,
Favoritenstraße 9–11, A-1040 Vienna, Austria
{oetsch,puehrer,tompits}@kr.tuwien.ac.at
[2] Johannes Kepler Universität Linz, Institut für Formale Modelle und Verifikation,
Altenbergerstraße 69, A-4040 Linz, Austria
Martina.Seidl@jku.at
[3] Technische Universität Wien, Institut für Softwaretechnik, 188/3
Favoritenstraße 9–11, A-1040 Vienna, Austria
[4] FTW Forschungszentrum Telekommunikation Wien GmbH
Donau-City-Straße 1, A-1220 Vienna, Austria
zwickl@ftw.at

Abstract. In the object-oriented world, much effort is spent into the development of dedicated tools to ease programming and to prevent programming errors. Recently, the techniques of *model-driven engineering* (MDE) have been proven especially valuable to manage the complexity of modern software systems during the software development process. In the world of answer-set programming (ASP), the situation is different. Much effort is invested into the development of efficient solvers, but the pragmatics of programming itself has not received much attention and more tool support to ease the actual programming phase would be desirable. To address this issue, we introduce the tool VIDEAS which graphically supports the partial specification of answer-set programs, applying technologies provided by MDE.

Keywords: answer-set programming, model-driven engineering, ER diagrams

1 Introduction

During the last decades, logic programming experienced a new impetus by the growth of answer-set programming (ASP) as one of the key technologies for declarative problem solving in the academic AI community. However, ASP could not attract the same interest as other programming languages outside academia so far. This lack of interest in ASP may be explained by the absence of a sufficiently supported software engineering methodology that could significantly ease the process of designing and developing ASP programs. Thus, more tool support is a declared aim of the ASP community. In particular, no modelling environment has been introduced in the context of developing

* This research has been partially supported by the the Austrian Science Fund (FWF) under grant P21698 and the Vienna Science and Technology Fund (WWTF) under grant ICT10-018.

answer-set programs that offers valuable abstraction and visualisation support during the development process. This absence of modelling tools may be explained by the fact that—in contrast to procedural programs—answer-set programs themselves are already defined at a high level of abstraction and may be regarded as executable specifications themselves. However, practice has shown that the development of answer-set programs is not always straightforward and that programs are, as all human-made artifacts, prone to errors. In fact, debugging in ASP is currently a quite active research field [1–9].

Consider for example the facts `airplan(boeing)` and `airplane(airbus).` This small program excerpt already contains a mistake. A predicate name is misspelled, which might result in some unexpected program behaviour. Furthermore, most current ASP solvers do not support type checking. A notable exception is the DLV$^+$ system [10] that supports typing and concepts from object-oriented programming. If values of predicate arguments are expected to come from a specific domain only, specific constraints have to be included in the program. This requires additional programming effort and could even be a further source for programming errors.

To support answer-set programmers, we developed the tool VIDEAS, standing for "VIsual DEsign support for Answer-Set programming", which graphically supports the partial specification of answer-set programs. Due to the close relationship between answer-set programs and deductive databases, the widely used *entity relationship diagram* (ER diagram) [11] is used as a starting point for the visualisation of answer-set programs. The constraints on the problem domain from an ER diagram are automatically translated to ASP itself. Having such constraints as part of a problem encoding can be compared to using assertions in C programs. To support the development of a fact base, VIDEAS automatically generates a program providing an input mask for correctly specifying the facts. To realise VIDEAS, we used well-established technologies from the active field of *model-driven engineering* (MDE) which provides tools for building the necessary graphical modelling editors as well as the code generator.

2 Answer-Set Programming with VIDEAS

We assume basic familiarity with ASP in what follows and refer to the literature for more details [12].

In object-oriented programming as well as in data engineering, it is common to model the necessary data structures by means of graphical models like UML class diagrams (CD) or the entity relationship diagram (ER diagram) [11]. In model-driven engineering (MDE) [13], such models serve as primary development artifacts from which code can be generated. Within the development process, models are more than mere documentation items as in traditional software engineering. Besides the fact that graphical visualisation is in general easier understandable for the human software engineer and programmer, models may be automatically transformed into executable code. Consequently, inconsistencies between the models and the code are impossible.

The VIDEAS system, whose basic functionality is described in what follows, is inspired by MDE principles and intended for graphically specifying the data model of an answer-set program by means of ER diagrams. From an ER diagram, certain constraints can be automatically derived to guide the development process and to support debug-

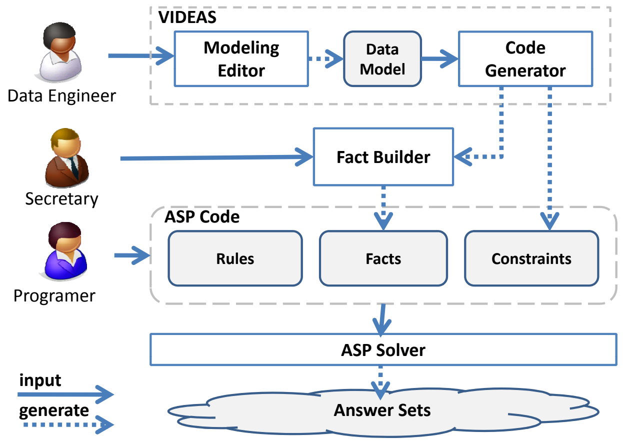

Fig. 1. The development process.

ging tasks. Similar approaches have been introduced in previous work [16, 17] where it is proposed to derive logic programs from *extended ER diagrams* (EER diagrams). In contrast to the VIDEAS approach, which aims at supporting the development of answer-set programs, the intention in these works was to provide a prototypical environment to experiment on various design approaches in order to reason about the instantiations of the EER diagrams. VisualASP [14] offers an environment for the graphical specification of answer-set programs by providing an editor for directly visualizing the ASP concepts. VIDEAS, in contrast, takes advantage of the abstraction power of the EER diagram and adopts the query by a diagram approach (cf. the survey article by Catarci et al. [15]) for program specification. The full potential of VIDEAS is exploited if it is integrated in a graphical development environment like the one due to Sureshkumar et al. [18].

An overview of the development process using VIDEAS is given in Fig. 1. In the VIDEAS framework, three tasks are necessary to build answer-set programs: (i) modelling, (ii) building a fact base, and (iii) implementing the program. The different tasks may be accomplished from people with different background. Specific knowledge on ASP is only required in the third step.

Modelling. In the first step, an ER diagram is specified using a graphical modelling editor that is part of the VIDEAS system (a screenshot of the editor is depicted in Fig. 2). The diagram describes entities and relations between entities of the problem domain under consideration. From the ER diagram, type and primary key constraints are derived which may be included in the final program for testing or debugging purposes. In particular, for every predicate P and each of its non-key attributes A, two rules are introduced that prohibit that two literals with predicate symbol P and different values for A sharing the same primary key are derived. Moreover, for each foreign key attribute, two rules are introduced ensuring that the key value references to an entity of the correct type. Fig. 3 presents a selection of the constraints covering the ER diagram in Fig. 2.

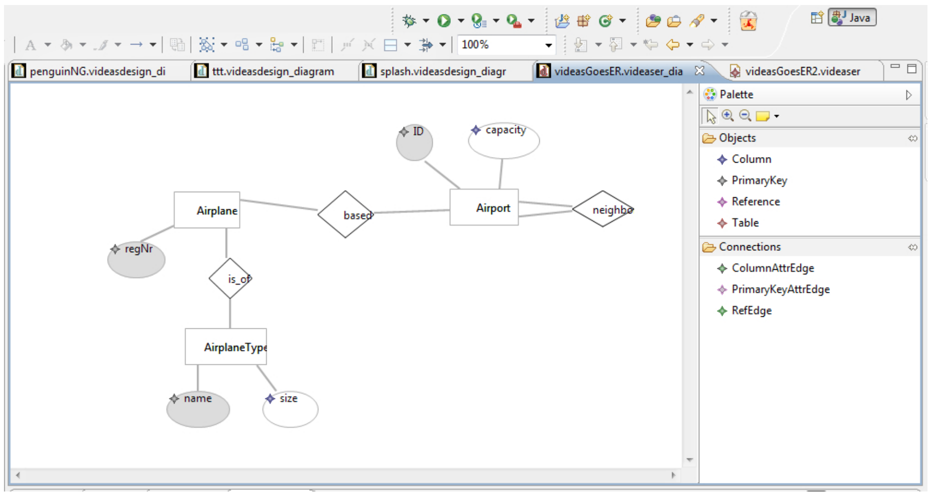

Fig. 2. Screenshot of the ER editor.

Building a fact base. After the modelling phase, the FactBuilder component allows to safely enter data by means of facts. The FactBuilder tool ensures that the entered data is consistent with the ER model. The resulting fact base may serve as an assertional knowledge base for the answer-set program. It is also possible to enter the data at a later point in time or to define multiple knowledge bases which increases the versatility of problem representations. Figure 4 gives an example exploiting the FactBuilder tool.

Implementation. Finally, the program under development has to be completed. That is, all properties of problem solutions beyond the constraints imposed by the ER diagram have to be formalised in ASP. VIDEAS does not impose any restriction on answer-set programmers concerning the implementation step but rather provides assistance for some parts of the development process by offering modelling and visualisation techniques as well as the automated generation of constraint systems.

3 Implementation

We next sketch how we developed a first prototype of VIDEAS based on standard model-engineering technologies. VIDEAS has been implemented on top of the Eclipse platform.[5] In particular, technologies provided by the Eclipse Modeling Framework (EMF)[6] and the Graphical Modeling Framework (GMF)[7] projects have been used. The meta-model representing the ER diagram modelling language has been created using the Ecore modelling language which is specified within the EMF project. Based on this Ecore model, a graphical editor has been created using GMF.

[5] `https://www.eclipse.org`.
[6] `http://www.eclipse.org/modeling/emf/`.
[7] `http://www.eclipse.org/modeling/gmf/`.

```
% PRIMARY KEY CONSTRAINT
nokPkAirportCapacity(ID) :- Airport(ID,CAPACITY1),
                            Airport(ID,CAPACITY2),
                            CAPACITY1 != CAPACITY2.
:- nokPkAirportCapacity(ID), Airport(ID,CAPACITY1).

% TYPE CONSTRAINTS
okAirplaneAirplaneTypeName(NAME) :-  Airplane(_,NAME,_),
                                     AirplaneType(NAME,_).
:-  not okAirplaneAirplaneTypeName(NAME), Airplane(_,NAME,_).
okAirplaneAirportID(ID) :- Airplane(_,_,ID), Airport(ID,_).
:-  not okAirplaneAirportID(ID), Airplane(_,_,ID).
```

Fig. 3. Excerpt of constraints generated from an ER diagram.

```
:add airplane
regNr: 1
airplaneType.name: Boeing737
airport.ID: ap1

% RESULTING FACT
airplane(1,Boeing737,ap1).
```

Fig. 4. An example for the FactBuilder component.

The code generator, which is implemented in Java, processes the models from the graphical editor. Again, this model is formulated in Ecore. The code generation itself can be grouped into three subsequent activities: First, the model is analysed. This allows to compute and to store the used literals based on the defined relationships, the chosen cardinalities, and the specified attributes. Second, type and primary key constraints are generated (cf. Fig. 3 for an example). Third, input forms are prompted which enable a developer to fill in values that are used for generating the facts of the program—the FactBuilder of VIDEAS (cf. Fig. 4). The FactBuilder component also implements features like the automated look-up of values from a known domain. Finally, the facts and constraints may be written to a file.

4 Conclusion and Future Work

The idea behind VIDEAS is to introduce successful techniques from model-driven engineering (MDE) to the ASP domain with the aim of supporting an answer-set programmer during a development phase. The distinguishing feature of MDE is that models are first-class citizens in the engineering process rather than mere documentation artifacts. In particular, programmers are encouraged to use ER diagrams to describe the data model of a problem domain before implementing a program. The benefit of an explicit model is that input masks for the consistent definition of a fact base for an answer-set program can be generated automatically. Furthermore, constraints represented by the ER model can be ported automatically into the language of ASP. Hence, consistency of any answer set with the data model of the ER diagram can always be guaranteed.

For future work, we intend to consider further concepts like inheritance relationship and other modelling languages like subsets of the UML class diagram as well. The UML class diagram may be particularly beneficial for ASP because the language-inherent extension mechanism of UML profiles may be used to adapt the UML class diagram to our specific purposes. We also plan to extend the VIDEAS framework to visualise potential inconsistencies between answer sets of a program and the data model directly at the level of the underlying ER diagram.

References

1. Brain, M., De Vos, M.: Debugging logic programs under the answer-set semantics. In: Proc. ASP'05. CEUR Workshop Proc., `CEUR-WS.org` (2005) 141–152
2. Syrjänen, T.: Debugging inconsistent answer set programs. In: Proc. NMR'06. (2006) 77–83
3. Brain, M., Gebser, M., Pührer, J., Schaub, T., Tompits, H., Woltran, S.: Debugging ASP programs by means of ASP. In: Proc. LPNMR'07. Springer (2007) 31–43
4. Mikitiuk, A., Moseley, E., Truszczynski, M.: Towards debugging of answer-set programs in the language PSpb. In: Proc. ICAI'07, CSREA Press (2007) 635–640
5. Caballero, R., García-Ruiz, Y., Sáenz-Pérez, F.: A Theoretical Framework for the Declarative Debugging of Datalog Programs. In: Proc. SDKB'08, Springer (2008), 143–159
6. Gebser, M., Pührer, J., Schaub, T., Tompits, H.: A meta-programming technique for debugging answer-set programs. In: Proc. AAAI'08, AAAI Press (2008) 448–453
7. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for logic programs under answer set semantics. Theory and Practice of Logic Programming **9**(1) (2009) 1–56
8. Wittocx, J., Vlaeminck, H., Denecker, M.: Debugging for model expansion. In: Proc. ICLP'09. Springer (2009) 296–311
9. Oetsch, J., Pührer, J., Tompits, H.: Catching the Ouroboros: On debugging non-ground answer-set programs. Theory and Practice of Logic Programming **10**(4-6) (2010) 513–529
10. Ricca, F., Leone, N.: Disjunctive logic programming with types and objects: The DLV$^+$ system. Journal of Applied Logic **5**(3) (2007) 545–573
11. Chen, P.: The entity-relationship model—Toward a unified view of data. ACM Transactions on Database Systems **1**(1) (1976) 9–36
12. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press, Cambridge, England (2003)
13. Schmidt, D.C.: Model-driven engineering. IEEE Computer **39**(2) (2006) 25–31
14. Febbraro, O., Reale, K., Ricca, F.: A Visual Interface for Drawing ASP Programs. In: Proc. CILC'10 (2010)
15. Catarci, T. and Costabile, M.F. and Levialdi, S. and Batini, C.: Visual query systems for databases: A survey. J. Visual Languages and Computing **8**(2) (1997) 215–260
16. Kehrer, N., Neumann, G.: An EER Prototyping Environment and its Implementation in a Datalog Language. In: Proc. ER'92. Volume 645 of LNCS, Springer (1992) 243–261
17. Amalfi, M., Provetti, A.: From extended entity-relationship schemata to illustrative instances. In: Proc. LID'08. (2008)
18. Sureshkumar, A., de Vos, M., Brain, M., Fitch, J.: APE: An AnsProlog Environment. In: Proc. SEA'07. (2007) 101-115