Integrating the ER Approach in an OO Environment*

M. Gogolla, R. Herzig, S. Conrad, G. Denker, and N. Vlachantonis
Technische Universität Braunschweig, Informatik, Abt. Datenbanken
Postfach 3329, D-38023 Braunschweig, Germany
e-mail: gogolla@idb.cs.tu-bs.de

Abstract

We translate Entity-Relationship (ER) schemas into the object-oriented specification language TROLL light. This language describes the Universe of Discourse (UoD) as a system of concurrently existing and interacting objects, i.e., an object community. Thereby two essential aspects, structure and behavior, are integrated in one formalism. By doing the translation from ER to TROLL light we preserve the visual advantages of the former and receive a formalism through the latter which can be mapped to an adequate object-oriented database system. Proceeding this way we hope our proposal for transforming ER schemas into TROLL light specifications provides a valuable link between structural and dynamic modeling.

1 Introduction

Nowadays the Entity-Relationship approach [Che76] has been accepted as a quasi standard [Teo90, BCN92] for the conceptual design of databases and information systems. This approach is undoubtedly a very popular tool for bridging the gap between database designers and users of database systems. One issue in favor of Entity-Relationship design is based on the fact that the graphical representation serves as an intuitively understandable platform for both database designers and users of database systems so that communication between them becomes easier. Another reason for the wide acceptance and application of this approach is due to the fact that an Entity-Relationship schema can be mapped in a systematic way to a relational database schema (see [EWH85, MMR86, TYF86, Hoh89], among others). But most approaches to Entity-Relationship modeling do not deal deeply with dynamic aspects, because the Entity-Relationship approach is used for modeling static structure though it ought not to be separated from the behavioral one. There are temporal Entity-Relationship extensions [KS91b, KS91a, HH91, RS91, Tau91] which add dynamic

*Work reported here has been partially supported by the CEC under Grant No. 6112 (COMPASS) and BMFT under Grant No. 01 IS 203 D (KoRSo).
aspects to the Entity-Relationship approach, but most of them are not directed to object-oriented databases.

Recently the advent of object-oriented based technology calls for and demands database design approaches and tools resulting in object-oriented database schemas. The main difficulties in tackling this problem arise from the lack of a common model and widely accepted principles. However, one advantage of object-oriented databases is that they cover both aspects, structure and behavior, and propose in this way an adequate formalism to describe the so-called Universe of Discourse (UoD), i.e., the part of the real world to be modelled. In this framework TROLL \textit{light} \cite{CGH92, GCH93} was developed as an object-oriented specification language describing the UoD as a system of concurrently existing and interacting objects, i.e., an object community. TROLL \textit{light} is a dialect of TROLL \cite{JSHSgl} and OBLOG \cite{SSE87}. Our language follows the decision that it is better to provide a modest number of basic constructs which are clear and easily applicable than to offer a lot of redundant language features. But because of the textual representation of TROLL \textit{light}, the language is in some respect less suitable to support the needs of a variety of users, like database designers and system users, in view of intuitive comprehensibility. Nevertheless TROLL \textit{light} is closer to object-oriented database systems since it provides a lot of language features capable of expressing behavioral aspects of systems. Our aim is to break the traditional way of translating Entity-Relationship schemas to relational databases. We translate Entity-Relationship schemas to TROLL \textit{light}, preserving the visual advantages of the former and receiving a formalism through the latter which can be mapped to an adequate object-oriented database system like this has been done in \cite{NCB91, Tar92}. After the structural design with the Entity-Relationship approach we suggest an object-oriented mapping to get on the way to an object-oriented schema.

Let us explain the TROLL \textit{light} approach in more detail. Our specification language comes along with an integrated, open development environment \cite{VHG+93}. The task of this environment is to give support for the creation of correct information systems on the basis of formal specifications. Some important ingredients of the environment are pictured in Figure 1. Rectangles (with upper left corner cut off) represent documents which are generated during the development process; ovals stand for tools which are invoked by the user, and arrows denote the main flow of information. We now look at the single tools item by item.

The \textbf{data type interpreter} supports the development of data type specifications. It allows to test the data type specification at hand and to validate that the specification meets the application requirements. The TROLL \textit{light} \textbf{editor} is understood to be a syntax-directed tool which also already checks context-sensitive conditions a TROLL \textit{light} specification has to fulfill. The TROLL \textit{light} \textbf{animator} is designed to simulate the specified object community. It provides object windows which enables the user for instance to observe attributes, to initiate state transitions by clicking events or to formulate queries against object states. By this the informal view of the real-world fragment to be modeled is validated against the current specification. The TROLL \textit{light} \textbf{prover} is given verification tasks by the user. It checks whether the desired properties are fulfilled by the specified TROLL \textit{light} templates. The TROLL \textit{light} \textbf{transformer} takes the specification of an object community and interactively generates executable code in an object-oriented programming language