Abstract. Object-oriented technology has become mature enough to satisfy many new requirements coming from areas like computer-aided design (CAD), computer-integrated manufacturing (CIM), or software engineering (SE). However, a competitive information management infrastructure often demands to merge data from CAD-, CIM-, or SE-systems with business data stored in a relational system. One approach for seamless integration of object-oriented and relational systems is to migrate from a relational to an object-oriented system. The first step in this migration process is reverse engineering of the legacy database. In this paper we propose a new graphical and executable language called Generic Fuzzy Reasoning Nets for modelling and applying reverse engineering knowledge. In particular, this language enables to define and analyse fuzzy knowledge which is usually all what is available when an existing database schema has to be reverse engineered into an object-oriented one. The analysis process is based on executing a fuzzy petri net which is parameterized with the fuzzy knowledge about a concrete database application.

1 Introduction and Related Work

Object-oriented technology has become mature enough to satisfy many new requirements coming from areas like computer-aided design (CAD), computer-integrated manufacturing (CIM), or software engineering (SE). Those new requirements are not fulfilled by relational technology [LS88, Mai89]. However, a competitive information management infrastructure often demands to merge data from CAD-, CIM-, or SE-systems with business data stored in a relational system. In addition, complex dependencies between those data stored in the different systems might exist and should be maintained. One approach for seamless integration of object-oriented and relational systems is to migrate the data (and the corresponding schema) from a relational to an object-oriented system.

The whole migration process consists of three steps which are (1) the schema migration process which maps a relational schema to an equivalent object oriented schema, (2)
the data migration process which converts extensions of the relational schema to extensions of the object oriented schema and (3) the application migration process which creates a new application program using the object oriented database for every application program that uses the legacy database.

This paper focuses on the first step in the migration process, namely the migration of the schema. A major job to do when migrating a schema is to analyse and reverse engineer the schema. This paper proposes a new approach for analysing existing relational schemas in order to reverse engineer them into object-oriented ones.

Analysing an existing database schema means to analyse the schema definition, the application code written in a so-called embedded SQL (Standard Query Language)-language where the embedding language is e.g. C or COBOL, and the available extensions of the schema. This analysis requires to express possibly uncertain assumptions about the schema which is to be reverse engineered. These assumptions may be partly deducible from the schema definitions and in particular the defined integrity constraints or from the SQL-queries within the application code. They may also only be deduced from the currently available schema extensions. Furthermore, an application has been developed incrementally over time by many developers who are sometimes even not accessible any more and who often did not do a good job on documentation. The result is that integrity constraints may even define contradicting constraints.

In order to retrieve an object-oriented schema an analysis tool should help to combine basic assumptions and to investigate their consequences. As many of those assumptions are uncertain, a tool should be able to deduce different possible consequences and their confidences. By supporting an incremental and interactive verification process, the tool should then let the reverse engineer decide finally which is the best alternative to choose.

Existing approaches [DA87,JK90,SK90,And94,PB94,PKBT94,FV95] do not support this kind of reasoning process, since the knowledge about the reverse engineering process is usually not defined explicitly but hard-coded in a batch-oriented analysis tool. One notable exception is an approach which is based on defining the reverse engineering knowledge in terms of PROLOG-rules [SLGC94]. However this approach as well as all others do not support the explicit (and thus easily changeable) definition and analysis of uncertain knowledge.

The next section gives a more detailed example which kind of semantic information we have to derive and which kind of uncertainty we have to deal with. Section 3 then presents our approach how to capture formally and precisely yet intuitively the (uncertain) reverse engineering knowledge about relational databases. Section 4 describes the inference engine which analyses basic facts about the investigated database and which then infers as much semantic information as possible by applying the reverse engineering knowledge described within Section 3. Within this step the inference engine incorporates and verifies interactively added user knowledge. Section 5 concludes with describing the current state of our work.