Human-Centered Reverse Engineering Environments should support Human Reasoning

Jens H. Jahnke, Jörg P. Wadsack

AG-Softwaretechnik, Fachbereich 17, Universität Paderborn,
Warburger Str. 100, D-33098 Paderborn, Germany;
e-mail: [jahnke|maroc]@uni-paderborn.de

1. Background and motivation

Today's information technology (IT) undergoes dramatic *mass changes* [McC98] due to urgent requirements like the coming of the next millennium (*Year-2000-problem*) [Mar97], the European currency union [Gro98], and emerging technologies like the *World Wide Web. Electronic Commerce* is about to become one of the key business technologies for the next decade. While new company start-ups are able to purchase modern information systems (IS) that meet these new requirements, longer established enterprises have to deal with pre-existing systems. In many cases, such *legacy* IS comprise complex architectures that have evolved over several generations of programmers and lack a sufficient technical documentation. Still, they maintain a great amount of valuable business data and their functionality is often critical for the mission of the enterprise. Consequently, a complete replacement of these systems is virtually impossible or at least implies a significant risk.

1.1 Computer aided reverse engineering - current limitations

In order to solve this problem, during the recent decade there has been increasing effort to develop methods and techniques to *reverse engineer* and modernize software legacies. In general, this is a complex task which requires a high amount of expert knowledge and coordination. *Computer-aided reverse engineering* (CARE) tools have a high potential to reduce this complexity and cope with the emerging mass change in IT. While *fully-automatic* approaches to CARE have proven useful to unburden the reengineer from a number of simple but laborious reverse engineering (RE) activities, it has been recognized that they are not sufficient to solve more complex RE problems [ALV93, Big90]. The reason for this insufficiency is the fact that one of the most valuable information sources in RE are humans. In many cases, developers, operators, and domain experts are able to contribute important knowledge about legacy systems.

As a consequence, many *interactive* CARE environments have been developed in industry and academy, e.g., [HEH+98, KWDE98, AG96, ONT96, MNS95, MWT94]. Such environments are often referred to be *human-centered*. Still, they are rarely applied by practitioners in real-world RE projects. This is mainly because of two significant limitations of current approaches, namely (1) they are not aware of the *mental model* of the tool user (*reengineer*) [JH98b] and (2) they prescribe a batch-oriented rather than an *evolutionary RE process*. The first problem reflects on the fact that many RE activities inherently deal with various heuristics which often lead to uncertain and inconsistent analysis results. The second problem considers that reengineers usually work in an explorative and iterative way to validate or refute intermediate analysis results and assumptions about the legacy system. We argue that human-centered CARE environments have to overcome these two limitations to increase their industrial acceptance. In this paper, we give a overview on an approach in the domain of database reverse engineering (DBRE) that aims to meet

this requirement. In the next section, we use a small RE sample scenario that deals with a *legacy database* (LDB) to motivate the importance of uncertain knowledge in RE processes. In Section 3, we introduce our approach to modeling this knowledge with a dedicated formalism called *Generic Fuzzy Reasoning Nets* (GFRN). Section 4 gives a short description of the mechanisms which are employed to execute GFRN models in an evolutionary and explorative RE process. Finally, Section 5 closes with some concluding remarks.

2. A reverse engineering sample scenario

In the following, we assume that the reader is familiar with the basic concepts of relational databases [EN94]. Figure 1 shows an example legacy database including small details of its procedural code, its database schema and the stored data. The recovered conceptual schema in EER-notation is depicted in Figure 2, while the semantic information which has to be deduced in order to reobtain this abstraction is shown in Figure 3.

Often, the schema definition of an LDB does not contain explicit definitions for foreign keys or candidate keys [EN94]. This is because of the limited functionality of antiquated database management systems. Possible keys can be found by searching the procedural code of the LDB for special patterns (sometimes called *clichés*). Examples for instances of clichés are the two SQL queries in the upper left part of Figure 1. The first cliché is called a *cyclic join* [And94]. It selects two entries in table `tenant` with the same value in column `house` but with different values in column `name`. This cliché serves as an indicator that tenants might be distinguishable by their name. On the other hand, the second query (*select distinct*) is an indicator against the assumption that `name` might be a key of table `tenant`: the column `house` of a tenant with a given `name` is selected, but the query contains the keyword `distinct`, which is used to avoid multiple elements with the same value in the result of a query. However, experience shows that the first cliché

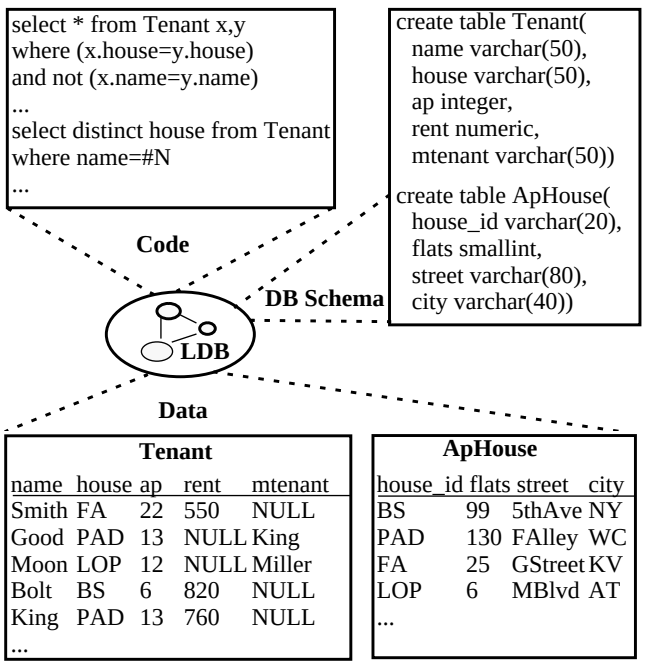

Figure 1. Details of a legacy database application

deserves a higher credibility than the second one. The assumption that name is a key might be disproved or supported by examining the data in table `tenant`. If there are two rows with the same value in column `name` the assumption must be refuted. On the other hand, if all rows have distinct values in column `name`, the assumed key could gain a greater credibility depending on the extent of the provided data in table `tenant`, e.g. if there are six hundred tenants with unique names there should be greater confidence that the assumption is true than if there are only six tenants.

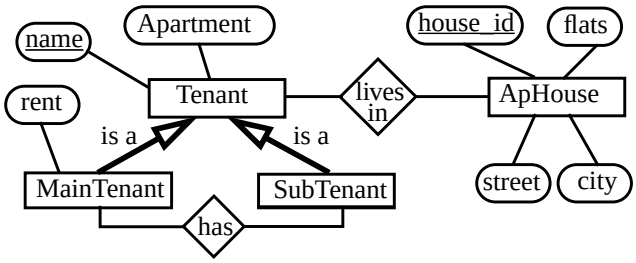

Figure 2. : Conceptual schema in EER-notation

In order to detect foreign keys between tables, the reengineer can search the application code for *join* clichés, similar to the first query in Figure 1. Further indications for a possible foreign key might be retrieved by checking the similarity of column names with other column names (e.g., columns `house` and `house_id`) or table names (e.g., column `mtenant` and table `tenant`). Obviously, columns which are supposed to have identical meaning should have the same type. For example columns `house` and `house_id` both are of type `varchar` but with a different length. However, as this occurs frequently even for columns with identical meaning, their types should be considered as compatible to a certain degree. Like before, possible foreign keys should be checked against the available data.

Keys:
Tenant(name)
ApHouse(house_id)
Equivalence Classes:
{Tenant.mtenant,Tenant.name}
{Tenant.house, ApHouse.house_id}
Foreign Keys:
Tenant.mtenant *references* Tenant.name
Tenant.house *references* ApHouse.house_id
Variants:
Tenant: MainTenant(name,house,ap,rent)
SubTenant(name,house,ap,mtenant)

Figure 3. Deduced Semantic Information

A further examination of the contents of table `tenant` shows that there seem to be two different variants of tenants: Each row in table `tenant` has either a NULL-value in column `rent` or in column `mtenant`. This reveals a hidden inheritance hierarchy. Again, the credibility of this indicator depends on the extent of the available data in table `tenant`.

3. Modeling uncertain RE knowledge

The above scenario exemplifies the importance of uncertain knowledge (heuristics) in the RE domain. In our approach, RE heuristics are modeled in a dedicated graphical formalism called *Generic Fuzzy Reasoning Nets* (GFRN). In the following, we give an informal introduction to this approach. For a formal definition of this formalism we refer to [JSZ97].

A GFRN is a graphical network of *fuzzy predicates* (with oval shape) and *implications* (represented as boxes) which are connected by arcs. Each implications has an associated *confidence value* (CV). Based on the theory of possibilistic logic [DLP94:PossibilisticLogic], the semantics of a CV is a lower bound of the necessity that the corresponding implication is valid. Arcs are labeled with formal parameters that can be used to specify constraints for implications. Figure 4 shows a detail of a GFRN which represents a part of the knowledge we used in our RE sample scenario, i.e., it aims to detect foreign keys in LDBs.

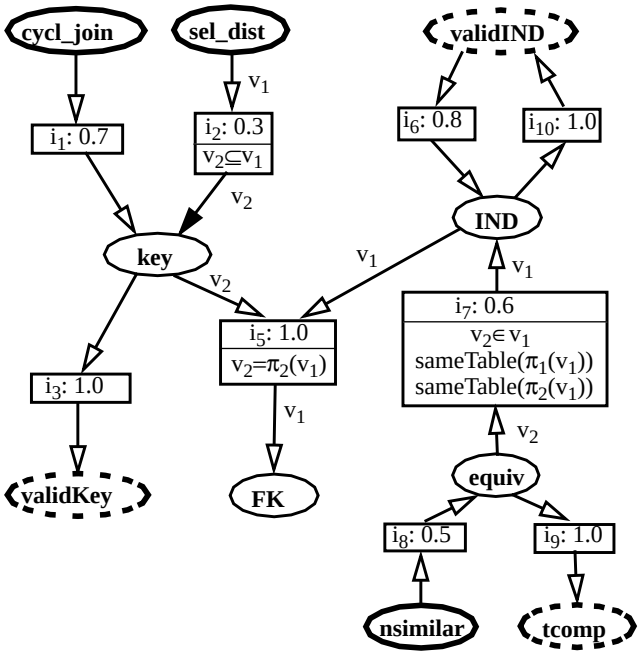

Figure 4. : Sample Generic Fuzzy Reasoning Net

Implication i_1 specifies the aforementioned heuristic that a *cyclic join* cliché is a rather credible indicator for a possible key candidate. On the other hand, implication i_2 models our experience that a *select-distinct* cliché over a set of columns v_1 serves as a negative indicator that subsets of v_1 might be keys. Implication i_3 specifies that an assumed key candidate may only be valid if there exists no counterexample in the data of the LDB.

A frequently used heuristic to detect relationships between tables, i.e., to detect equivalent columns in different tables, is to check column names for similarities. Obviously, it is necessary that columns which have identical meaning are type compatible. This heuristic is modeled with Implications i_8 and i_9. Implication i_7 is an example that even complex rules can be expressed in GFRN specifications. It specifies

that pairs of equivalent columns in two different tables are an indicator for an inclusion dependency (IND) over these columns. At this, Parameter v_1 is constrained to be a set of pairs v_2. `sameTable` is defined to be a boolean function that evaluates to *true* iff all columns in its argument are in the same table. Finally, Functions π_1 and π_2 represent the relational projection on the first and the second element of each pair in v_1, respectively.

Again, the available data has to be checked whether or not the supposed INDs can be disproved (Implication i_{10}). Implication i_6 specifies that the validation of an assumed IND over a huge amount of available data may further support this assumption. Finally, the presence of a foreign key can be deduced from the existence of an IND over pairs of columns, where the second element of each pair must constitute a key candidate (Implication i_5).

3.1 The role of automatic analysis operations

Some fuzzy propositions can be determined by automatic analysis operations, e.g., checking for similar column names, comparing column types, validating assumptions against the available data, searching procedural code for clichés, etc. In the GFRN formalism, such analysis operations can be bound to fuzzy predicates. Depending on the point of time, when these operations are performed, the corresponding predicates are called either *data-driven* or *goal-driven*. In Figure 4, data-driven predicates are represented by bold ovals while goal-driven predicates have a dashed shape.

Analysis operations which have been bound to data-driven predicates are performed at the *beginning* of the analysis process, to deliver initial information about the LDB under investigation. As an example, the data-driven predicate `nsimilar` is bound to an operation that compares the names of different columns according to a possibility function as a measure of similarity. Such a possibility function could for example be based on string similarity metrics like the Levenshtein-distance [AWY90].

In contrast to data-driven predicates, analysis operations of goal-driven predicates are invoked on-demand *during* the analysis process to refute or support intermediate results. The validation of an IND via the available data (`validIND`) is an example for an expensive operation that should only be performed for INDs which have already been indicated.

4. Achieving an evolutionary RE process

We have chosen a *fuzzy petri net* (FPN) [JH98a] as an inference engine to execute GFRN models as it allows for non-monotonic reasoning and, thus, facilitates the desired evolutionary RE process. This means that each (fuzzy) fact in an RE project is represented by one *place* in the FPN, while implications are mapped to *transitions*. The particular FPN model used in our approach is an extension of the FPN model described in [KM96]. The major difference compared to the original FPN model is that our approach deals with inconsistent knowledge. We refer to [JH98a] for a detailed definition of the employed FPN model.

The actual analysis process is illustrated in Figure 5. It starts by executing all automatic operations which have been assigned to data-driven predicates in the GFRN model. This first investigation delivers the initial amount of facts about the subject LDB. At this early stage, the semantic information inferred by the inference engine is likely to be incomplete and inconsistent. These inconsistencies have to be resolved by the reengineer, who can initiate further investigations, discuss intermediate analysis results with domain experts and developers, and enter additional assumptions.

The inference process is resumed as soon as new information about the LDB becomes available. During each iteration, automatic (goal-driven) analysis operations are performed on-demand to refute or support assumptions. This evolutionary RE process is continued until the produced semantic information is consistent and complete.

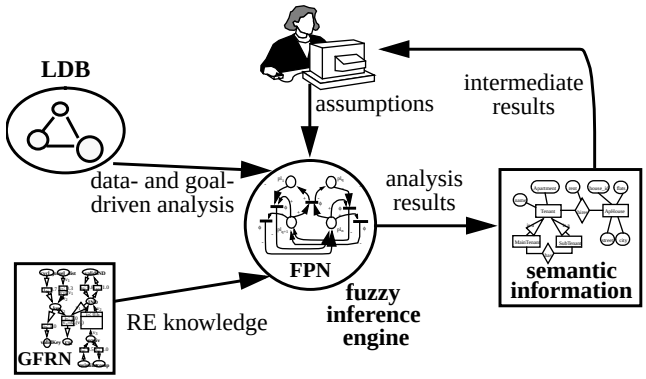

Figure 5. The supported evolutionary RE process

5. Concluding remarks

It has been commonly recognized that RE is a complex and human-intensive process [ALV93, Big90]. CARE environments have a great potential to decrease this complexity and shorten RE project durations. However, we argue that in order to be accepted in industry, user-centered CARE environments have to be aware of the explorative and evolutionary nature of RE processes. They have to cope with uncertain and inconsistent information about legacy systems. In this paper, we give an overview on our approach to tackle these problems in the domain of DBRE. The described concepts have been implemented in the *Varlet* CARE environment [JSZ96]. Currently, we have started to apply this environment to practical examples in industrial projects. First experiences show that our approach is feasible and promising. Our current research focus is on employing learning algorithms of connectionist systems to adapt the credibilities of specified RE heuristics to changing application contexts [JS99]. Furthermore, we aim to apply our results to other areas in RE, e.g., to recover the design of object oriented software [JZ97].

References

[AG96] D. C. Atkinson and W. G. Griswold. The design of whole-program analysis tools. In *Proc. of the 18th Int. Conf. on Software Engineering, Berlin, Germany*, pages 16–27, 1996.

[ALV93] F. Abbattista, F. Lanubile, and G. Visaggio. Recovering conceptual data models is human-intensive. In *Proc. of 5th Intl. Conf. on Software Engineering and Knowledge Engineering, San Francisco, California, USA*, pages 534–543, 1993.

[And94] M. Andersson. Extracting an Entity Relationship Schema from a Relational Database through Reverse Engineering. In *Proc. of the 13th Int. Conference of the Entity Relationship Approach, Manchester*, pages 403–419. Springer, 1994.

[AWY90] L. Allison, C. S. Wallace, and C. N. Yee. When is a string like a string? *AI & Maths.*, 1990.

[Big90] Ted J. Biggerstaff. Human-oriented conceptual abstractions in the reengineering of software. In *Proceedings of the 12th International Conference on Software Engineering*, page 120, March 1990.

[DLP94] D. Dubois, J. Lang, and H. Prade. *Possibilistic Logic*, pages 439–503. Clarendon Press, Oxford, 1994.

[EN94] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Benjamin/ Cummings, Redwood City, 2 edition, 1994.

[Gro98] Kurt Grotenhuis. Crossing the euro rubicon. *IEEE Spectrum*, 35(10):30–33, October 1998.

[HEH+98] J. Henrard, V. Englebert, J.-M. Hick, D. Roland, and J.-L. Hainaut. Program understanding in database reverse engineering. Technical Report RP-98-004, Institute d'Informatique, University of Namur, Belgium, 1998.

[JH98a] J. H. Jahnke and M. Heitbreder. Design recovery of legacy database applications based on possibilistic reasoning. In *Proceedings of 7th IEEE Int. Conf. of Fuzzy Systems (FUZZ'98). Anchorage, USA.*. IEEE Computer Society, May 1998.

[JH98b] Stan Jarzabek and Riri Huang. The case for user-centered case tools. *Communications of the ACM*, 41(8):93–99, August 1998.

[JS99] Jens H. Jahnke and Christoph Strebin. Adaptive tool support for database reverse engineering. In *Proc. of 1999 Conference of the North American Fuzzy Information Processing Society, New York, USA*, June 1999. submitted.

[JSZ96] J. H. Jahnke, W. Schäfer, and A. Zündorf. A design environment for migrating relational to object oriented database systems. In *Proc. of the 1996 Int. Conference on Software Maintenance (ICSM'96)*. IEEE Computer Society, 1996.

[JSZ97] J. H. Jahnke, W. Schäfer, and A. Zündorf. Generic fuzzy reasoning nets as a basis for reverse engineering relational database applications. In *Proc. of European Software Engineering Conference (ESEC/FSE)*, number 1302 in LNCS. Springer, September 1997.

[JZ97] Jens Jahnke and Albert Zundorf. Rewriting poor design patterns by good design patterns. In Serge Demeyer and Harald Gall, editors, *Proceedings of the ESEC/FSE Workshop on Object-Oriented Re-engineering*. Technical University of Vienna, Information Systems Institute, Distributed Systems Group, September 1997. Technical Report TUV-1841-97-10.

[KM96] A. Konar and A. K. Mandal. Uncertainty management in expert systems using fuzzy petri nets. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):96–105, February 1996.

[KWDE98] B. Kullbach, A. Winter, P. Dahm, and J. Ebert. Program comprehension in multi-language systems. In *Working Conference on Reverse Engineering*, pages 135–143, Hawai, USA, October 1998. IEEE Computer Society, IEEE Computer Society Press.

[Mar97] Robert A. Martin. Dealing with dates: Solutions for the year 2000. *Computer*, 30(3):44–51, March 1997.

[McC98] Thomas J. McCabe. Does reverse engineering have a future? Keynote of the 5th Working Conference on Reverse Engineering, Honolulu, Hawaii, USA, October 1998.

[MNS95] Gail C. Murphy, David Notkin, and Kevin Sullivan. Software Reflexion Models: Bridging the Gap between Source and High-Level Models. In *Proceedings of SIGSOFT'95 Third ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 18–28, October 1995.

[MWT94] Hausi A. Müller, Kenny Wong, and Scott R. Tilley. Understanding software systems using reverse engineering technology. In *Proceedings of the 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences (ACFAS '94)*, pages 41–48, Montreal, PQ, "16–17 "# may 1994.

[ONT96] ONTOS Inc., hree Burlington Woods, Burlington, MA, USA. *ONTOS Object Integration Server for Relational Databases 2.0 - Schema Mapper User's Guide*, 2.0 edition, 1996.