Varlet: Human-Centered Tool Support for Database Reengineering

(Extended Abstract)

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

Software evolution and maintenance problems might be caused by all kinds of new or changed requirements. However, McCabe [McC98] has identified a number of requirements, which are currently of special importance because they are responsible for significant *mass changes* in today's business software. Among these central requirements are the *Year-2000* problem [Mar97a], the *Euro-conversion* problem [Gro98], and the ability to compete on a global, electronic market. The primary concern of all these requirements is the issue of how business data should adequately be represented in software systems. The addressed problems range from simple questions, e.g., for the number of digits that are necessary to represent a date (Year-2000 problem), up to complex architectural decisions, e.g., how to federate data maintained by diverse (formerly autonomous) information systems and integrate these systems with the Web to facilitate electronic commerce.

If a *legacy software system* (LSS) has to be adapted to one of these requirements, a conceptual documentation of its data structure (DS) is thus often a necessary prerequisite to achieve the maintenance goal. Moreover, a conceptual DS is an excellent starting point for the migration to modern programming languages, as they are usually data-oriented [GK93]. This is because the conceptual DS reflects major business rules but is fairly independent from procedural application code.

The importance of a sound understanding of legacy DS in maintenance and migration projects has been pointed out by several researchers and practitioners [Aik95,HEH+98,GK93]. Unfortunately, a corresponding documentation is missing, obsolete, or inconsistent for many existing LSS. The process of recovering such a documentation from a subject LSS is called *data reverse engineering* (DRvE) [Aik95]. In the case that some kind of database management system (DBMS) is used as a platform for the LSS, this process is also more specifically called *database reverse engineering* (DBRvE) respectively *database reengineering* (DBRE) if a subsequent modification of the LSS is considered.

According to [Aik95], DBRvE processes are in general more structured than arbitrary DRvE processes. Consequently, the potential for tool support and automation is much higher in DBRE. The main reason for this is that the used DBMS already provides the reengineer with some basic information about the implemented physical data structure in form of a *schema catalog*. Still, important structural and semantic information about the data structure might not be explicit but indicators for this information might be found in different parts of the legacy database (LDB), including its procedural code, stored data, and obsolete documentation. Moreover, domain experts and developers might be able to contribute further valuable information about the LDB. The DBRE problem is to find, weigh, and merge these indicators to yield a consistent conceptual DS (cf. Figure 1). In many cases heuristics and uncertain expert knowledge have to be employed.

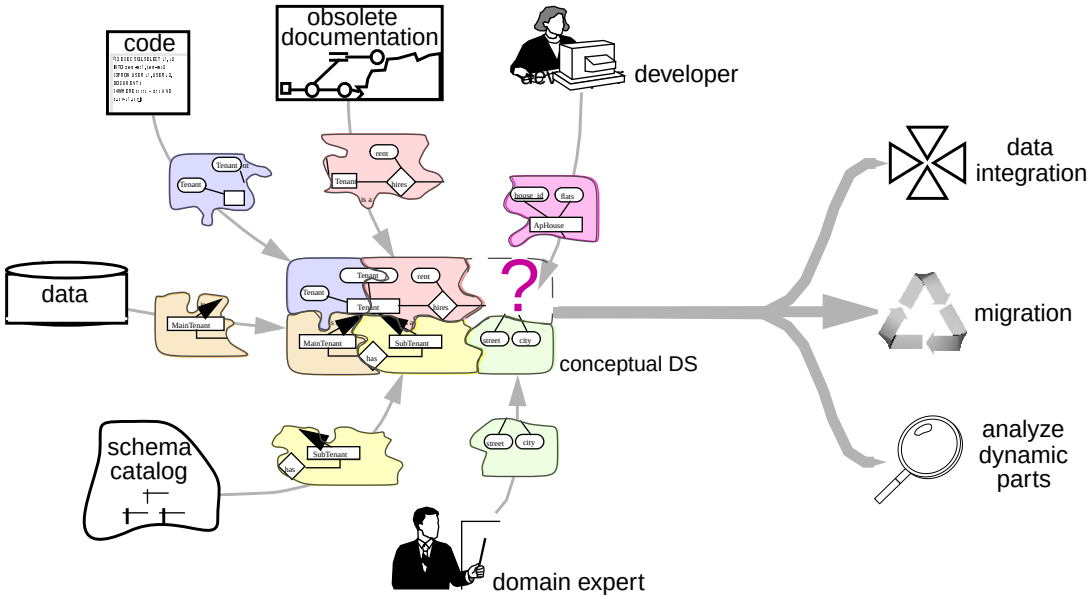

Figure 1. Conceptual DS as a starting point for subsequent RE activities

2 Limitations of current DBRE tools

In the last two decades, many researchers have developed concepts and techniques for automating certain DBRE activities, in order to reduce the complexity of the DBRE task [vdBKV97]. Many of these approaches have been implemented in computer-aided reengineering (CARE) tools and some of them have proven to be useful for practical applications. CARE tools seem to have great potential to assist the reengineer, e.g., by performing laborious analysis steps, browsing information about legacy software artifacts, and guiding the DBRE process. However, such tools are rarely used in industry [Sto98, p. 3]. Researchers and practitioners have identified the most significant reasons for this as their lack of *customizability* [MNB+94] and *human-awareness* [JH98b, Sto98, JW99a]. Furthermore, they do not allow for *incremental* and *iterative* DBRE processes [WSK97].

2.1 Lack of customizability

Customizability is a crucial requirement on CARE tools, because LDBs are different with respect to many technical and non-technical parameters. They are based on diverse (old) hard- and software platforms, use miscellaneous programming languages, contain various optimization structures and arcane coding concepts (*idiosyncrasies* [BP95, HHEH96]), and comprise different naming conventions. Furthermore, DBRE projects may be driven by a great variety of different goals. Such goals range from fixing defects (e.g., Year-2000-Problem [Mar97b]), over extending or integrating data structures, up to completely changing the architecture of an LDB, e.g., migrating from a procedural, monolithic, and autonomous legacy system to an open, distributed, and object-oriented application.

Most existing CARE tools employ general-purpose programming languages to implement DBRE heuristics, analysis operations, and processes. As a consequence, these environments can hardly be customized for changing application contexts. Some more advanced approaches aim to tackle this problem by providing application programming interfaces (APIs) [BM98] or interpreters for scripting languages [Rat98,TWSM94]. Such interfaces offer the flexibility to adapt the CARE tool with less effort or even without the need for

recompilation. Still, a limitation of these approaches is their low level of abstraction: aspects like DBRE heuristics and processes have to be programmed in form of procedural scripts, even though they would be more adequately described in a declarative formalism, e.g., in form of rules [HK94, PS92] or graphical networks [DG95, Loo88]. Other, mostly generative approaches employ a more abstract specification of the legacy target platform (e.g., [KWDE98, MCAH95, Jar95, PP94]). They have proven useful for fully-automatic activities like program pattern recognition [QYW98], schema transformation [MCAH95], and code restructuring [SV98]. However, a CARE tool should also facilitate the customization of DBRE heuristics (e.g., [SLGC94]) and the performed process (e.g., [HK94]). Moreover, it should provide an *open architecture*, i.e., it should allow the integration of other CARE tools (e.g., analyzers, extractors, and transformers).

2.2 Lack of human-awareness

One of the most valuable information sources in DBRE are humans. Developers, operators, and domain experts might be able to contribute important knowledge about a subject LDB. Hence, CARE tools should be *human-aware*, i.e., they should consider human knowledge and interaction in the supported DBRE process. The human-awareness of existing CARE tools can be characterized according to two main aspects. The first aspect regards the *role* of human knowledge in the DBRE process, while the second aspect regards its *representation*.

A comparison of CARE tools according to the role of human knowledge concerns the question: *at which point in the supported DBRE process is human knowledge considered?* We classify existing approaches as either *human-excluded*, *human-involved*, or *human-centered* (cf. Figure 2).

Human-excluded CARE tools perform fully-automatic analysis and conceptual translation operations on a subject LSS (e.g., Hüs98, BGD97, Fon97, RH97, FV95, MCAH95, MN95, SLGC94, Wil94, RHSR94]). As an output, they produce (a number of) analysis reports that can be used as a starting point for a manual semantic abstraction and redesign activity. Human knowledge and intervention is not considered in these batch-oriented tool processes.

Many CARE tools involve humans in partly interactive DBRE processes. Such approaches usually start with an automatic analysis of the LSS in order to extract important structural information. Based on the analysis results, the user can subsequently explore the LSS, and interactively add further semantic abstractions. Examples for such more sophisticated approaches are [HEH$^+$98, KWDE98, Hol98, LO98, Nov97, FHK$^+$97, ONT96, SM95, MAJ94, AL94, MWT94]. We call these tools *human-involved* as opposed to *human-centered*, because human knowledge is only considered in the second stage of the supported DBRE process (*completion/redesign*, cf. Figure 2). Finally, we denote CARE tools as *human-centered*, if they enable interactive DBRE processes including both kinds of activities, *software analysis* and *abstraction/redesign*, e.g., [HEH$^+$98, AG96, MNS95].

The second aspect of *human-awareness* regards the way how human knowledge is represented in CARE environments. DBRvE activities deal with various heuristics that deliver uncertain analysis results and reengineers have uncertain assumptions about the internal realization of LDBs. Existing CARE tools do not consider this human mental model and represent assumptions and analysis results without a measure for their confidence. Furthermore, DBRvE activities generally have an evolutionary and explorative nature. It frequently occurrs that heuristics deliver inconsistent analysis result, i.e., the reengineer discovers indicators in favor of a given assumption as well as against it. Current CARE tools do not tolerate such inconsistencies and most of them do not even indicate them. This is a severe limitation because in a later stage of the DBRvE process it might become clear that the hypothesis that has been chosen in such a situation has to be refuted. In this case, the knowledge about the indication of its alternative has been lost due to the inability to represent “both sides of the coin”.

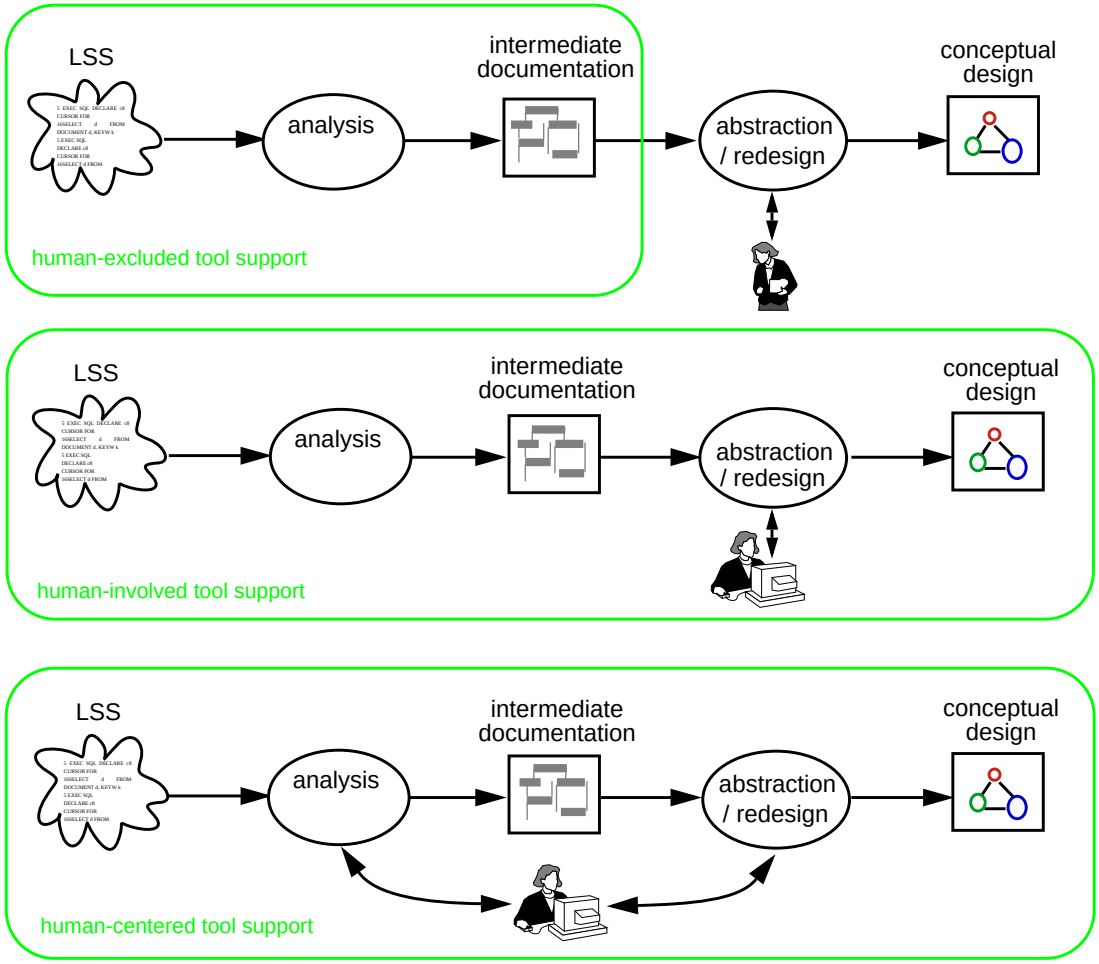

Figure 2. CARE tool classification according to the role of human knowledge

2.3 Insufficient support for iterations

Another problem of currently existing CARE tools is their limited support for process iterations. They usually assume that the process of knowledge accumulation is *monotonic* and prescribe a strictly phase-oriented methodology. In practice this is an important limitation, as iterations between analysis and abstraction steps occur frequently: When a reengineer learns more about the abstract design of an LSS, (s)he often refutes some initial assumptions or does some further investigations. For example, as soon as an intermediate abstraction of an LSS has been created it can be discussed with domain experts which might elicit additional information. In many cases, this new information contradicts to some initial assumptions. Strictly phase-oriented tools do not aid the reengineer in detecting and resolving such inconsistencies. In case of iterations to early analysis activities the reengineer loses the work (s)he has performed interactively in later abstraction and redesign activities.

3 The *Varlet* approach

We adopt a process that consists of two main phases, namely *schema analysis* and *conceptual abstraction* (and migration). In the first phase, the different parts of the LDB are analyzed to obtain a consistent and complete logical data structure (logical DS) for the implemented physical schema. In the second phase (abstraction), this logical DS is transformed into a conceptual DS that is the basis for subsequent modification

or migration activities. In this dissertation, we develop concepts and techniques that allow to build CARE environments which overcome the aforementioned limitations of current approaches in the DBRE domain.

3.1 GFRN to achieve customizability

We propose a dedicated graphical language named *Generic Fuzzy Reasoning Nets* (GFRN) to customize CARE tools according to their specific application context [JSZ97,JZ97,JH98a,JS99]. GFRN specifications separate declarative knowledge from operational aspects. They provide a high level of abstraction and extensibility. Analysis operations that have been developed in other DBRE approaches can easily be integrated with GFRN specifications. We develop a prototype CARE environment that is parameterized by GFRN specifications and includes a customization front-end for this purpose.

We reflect the mental model of the reengineer by representing DBRvE knowledge in the framework of possibility theory [DLP94]. This approach allows to deal with uncertain and inconsistent analysis results. We develop a non-monotonic *inference engine* (IE) that supports the reengineer in his/her DBRvE activities by propagating and indicating measures of credibility and inconsistency. For this purpose the IE interprets the declarative knowledge that is specified in the current GFRN specification. In addition, the IE is also capable of executing the analysis operations that are specified in the GFRN. This is done automatically during the DBRE process to search for indicators or validate intermediate hypotheses. With this approach, we obtain a CARE tool that plays a more *active* role in the DBRE process than existing tools. A graphical *dialog component* visualizes the current knowledge about the persistent structure of an LDB to the user. This component provides powerful abstraction and query mechanisms to focus the reengineers attention on the most controversial parts of the legacy schema. It enables the reengineer to enter the results of manual investigations or add new hypotheses that might be falsified or supported by the IE. Hence, our approach intertwines automatic and manual analysis activities in an explorative and evolutionary process that is guided by the IE until a consistent (and definite) logical DS is obtained.

3.2 Incremental consistency management

We apply *graph grammars* [Ros97] to map the analyzed logical DS into a conceptual (object-oriented) data model. The resulting conceptual DS can interactively be enhanced and redesigned to exploit additional abstraction mechanisms and migrate to new requirements. The available redesign operations are formally defined by graph transformations. Based on this formalization we develop a consistency management component that incrementally propagates modifications of the logical DS to the (redesigned) conceptual DS in case of process iterations [JSZ97,JW99b,JZ99,JZ98]. This unburdens the reengineer from the error-prone and time-consuming task to determine such inconsistencies manually. The developed consistency management component can be viewed as an adaption of general techniques described in [Nag96] to the DBRE domain.

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.

[Aik95] P. Aiken. *Data Reverse Engineering: Slaying the Legacy Dragon*. McGraw-Hill, 1995. This is the first book describing the process of recovering data architectures from existing information systems and using it to develop a foundation for enterprise integration and other reengineering efforts.

[AL94] Daniel Aebi and Reto Largo. Methods and tools for data value re-engineering. In *Applications of Databases (ADB-94)*, volume 819 of *LNCS*, pages 400–411. Springer, June 1994.

[BGD97] Andreas Behm, Andreas Geppert, and Klaus R. Dittrich. On the migration of relational schemas and data to object-oriented database systems. In *Proc. 5th International Conference on Re-Technologies for Information*

Systems, Klagenfurt, Austria, December 1997.

[BM98] Elisa Baniassad and Gail Murphy. Conceptual module querying for software reengineering. In *Proceedings of the 20th International Conference on Software Engineering*, pages 64–73. IEEE Computer Society Press, April 1998.

[BP95] Michael Blaha and William Premerlani. Observed idiosyncracies of relational database designs. In *Second Working Conference on Reverse Engineering*. IEEE, 1995.

[DG95] W. Deiters and V. Gruhn. Process management in practice - applying the FUNSOFT net approach to large scale processes. In *icse17*, Seattle, Washington, US, September 1995. submitted for publication.

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

[FHK+97] P. J. Finnigan, R. C. Holt, I. Kalas, S. Kerr, K. Kontogiannis, H. A. Müller, J. Mylopoulos, S. G. Perelgut, M. Stanley, and K. Wong. The software bookshelf. *IBM Systems Journal*, 36(4):564–??, 1997.

[Fon97] J. Fong. Converting relational to object-oriented databases. *ACM SIGMOD Record*, 26(1), March 1997.

[FV95] C. Fahrner and G. Vossen. Transforming Relational Database Schemas into Object-Oriented Schemas according to ODMG-93. In *Proc. of the 4th Int. Conf. of on Deductive and Object-Oriented Databases 1995*, 1995.

[GK93] Harald Gall and Ren'e Klösch. Capsule oriented reverse engineering for software reuse. In *Proceedings of the European Conference on Software Engineering 1993*, pages 418–433, 1993.

[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.

[HHEH96] J.-L. Hainaut, J.-M. Hick, V. Englebert, and J. Henrard. Understanding the implementation of IS-A relations. *Lecture Notes in Computer Science*, 1157:42–??, 1996.

[HK94] George T. Heineman and Gail E. Kaiser. Incremental process support for code reengineering. In *Proceedings of the International Conference on Software Maintenance 1994*, pages 282–290. IEEE Computer Society Press, September 1994.

[Hol98] R.C. Holt. Structural manipulations of software architecture using tarski relational algebra. In *Working Conference on Reverse Engineering*, pages 210–219, Hawai, USA, October 1998. IEEE Computer Society, IEEE Computer Society Press.

[Hüs98] F. Hüsemann. Eine erweiterte Schemaabbildungskomponente für Datenbank–Gateways. In *10. Workshop "'Grundlagen von Datenbanken"'*, pages 52–56, Konstanz, June 1998. Konstanzer Schriften in Mathematik und Informatik Nr. 63, Universität Konstanz.

[Jar95] Stan Jarzabek. Pqtl: a language for specifying program transformations. In *Proc. of Intl. Conf. on Software Engineering, Seattle, USA.*, 1995.

[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.

[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.

[JW99a] Jens H. Jahnke and Jörg Wadsack. Human-centered reverse engineering environments should support human reasoning. In *Proc. of the 1st International Workshop on Soft Computing Applied to Software Engineering (SCASE'99). Limerick, Ireland*, April 1999.

[JW99b] Jens H. Jahnke and Jörg Wadsack. Integration of analysis and redesign activities in information system reengineering. In *Proc. of the 3rd European Conference on Software Maintenance and Reengineering (CSMR'99). Amsterdam, NL*, March 1999.

[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.

[JZ98] J. H. Jahnke and A. Zündorf. Using graph grammars for building the varlet database reverse engineering environment. In *Proc. of Theory and Application of Graph Transformations, Paderborn, Germany*, LNCS. Springer Verlag, Berlin, November 1998. to appear.

[JZ99] Jens H. Jahnke and A. Zündorf. *Handbook of Graph Grammars and Computing by Graph Transformation - Application*, volume 2, chapter Applying Graph Transformations To Database Re-Engineering. World Scientific, Singapore, 1999. to appear.

[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.

[LO98] T. Lin and L. O'Brian. Fepss: A flexible and extensible program comprehension support system. In *Working Conference on Reverse Engineering*, pages 40–49, Hawai, USA, October 1998. IEEE Computer Society, IEEE Computer Society Press.

[Loo88] C. G. Looney. Fuzzy petri nets for rule-based decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):178–183, February 1988.

[MAJ94] U. A. Johnen M. A. Jeusfeld. An executable meta model for re-engineering of database schemas. Technical Report 94-19, Technical University of Aachen, Germany, 1994.

[Mar97a] Robert A. Martin. Dealing with dates: Solutions for the year 2000. *Computer*, 30(3):44–51, March 1997.

[Mar97b] Robert A. Martin. Dealing with dates: Solutions for the year 2000. *Computer*, 30(3):44–51, March 1997.

[MCAH95] P. Martin, J. R. Cordy, and R. Abu-Hamdeh. Information capacity preserving of relational schemas using structural transformation. Technical Report ISSN 0836-0227-95-392, Dept. of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, November 1995.

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

[MN95] Gail C. Murphy and David Notkin. Lightweight source model extraction. In *Proc. of ACM SIGSOFT Symposium on the Foundations of Software Engineering*, October 1995.

[MNB+94] L. Markosian, P. Newcomb, R. Brand, S. Burson, and T. Kitzmiller. Using an enabling technology to reengineer legacy systems. *Communications of the ACM*, 37(5):58–70, May 1994.

[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.

[Nag96] M. Nagl, editor. *Building tightly integrated software development environments*, volume 1170 of *LNCS*. Springer, Berlin, 1996.

[Nov97] Novera Software Inc., 3 Burlington Woods, Burlington, MA 01830, USA. *Novera EPIC Database Builder (TM)*, release 1.3 edition, September 1997.

[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.

[PP94] S. Paul and A. Prakash. A framework for source code search using program patterns. *IEEE Transactions on Software Engineering*, 20(6):463–475, June 1994.

[PS92] B. Peuschel and W. Sch"afer. Concepts and Implementation of a Rule-based Process Engine. In *Proc. of the 14th Int. Conf. on Software Engineering, Melbourne, Australia*, pages 262–279. IEEE Computer Society Press, 1992.

[QYW98] Alex Quilici, Qiang Yang, and Steven Woods. Applying plan recognition algorithms to program understanding. *Journal of Automated Software Engineering*, 5(3):1–26, 1998.

[Rat98] Rational Software Corp., 18880 Homestead Road, Cupertino, CA 95014, USA. *Rational Rose 98 - Using Rational Rose / Oracle 8*, 1998.

[RH97] S. Ramanathan and J. Hodges. Extraction of object-oriented structures from existing relational databases. *ACM SIGMOD Record*, 26(1), March 1997.

[RHSR94] Thomas Reps, Susan Horwitz, Mooly Sagiv, and Genevieve Rosay. Speeding up slicing. In *Proc. of ACM*

SIGSOFT, New Orleans LA, USA., December 1994.

[Ros97] Grzegorz Rosenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific, Singapore, 1997.

[SLGC94] O. Signore, M. Loffredo, M. Gregori, and M. Cima. Reconstruction of er schema from database applications: a cognitive approach. In *Proc. of 13th Int. Conference of ERA, Manchester*, pages 387–402. Springer, 1994.

[SM95] M-A D Storey and H. Mueller. Manipulating and documenting software structures using SHriMP views. In *International Conference in Software Maintenance*, pages 275–285. IEEE Computer Society Press, 1995.

[Sto98] M. A. D. Storey. *A Cognitive Framework for Describing and Evaluating Software Exploration Tools*. PhD thesis, Simon Fraser University, Vancouver, B.C., Canada, 1998.

[SV98] Alex Sellink and Chris Verhoef. Native patterns. In *Working Conference on Reverse Engineering*, pages 89–103, Hawai, USA, October 1998. IEEE Computer Society, IEEE Computer Society Press.

[TWSM94] S. R. Tilley, K. Wong, M-A. D. Storey, and H. A. Müller. Programmable reverse engineering. *International Journal of Software Engineering and Knowledge Engineering*, 4(4):501–520, 1994.

[vdBKV97] Mark van den Brand, Paul Klint, and Chris Verhoef. Reverse engineering and system renovation: an annotated bibliography. *ACM Software Engineering Notes*, 22(1), January 1997.

[Wil94] Linda M. Wills. Using attributed flow graph parsing to recognize programs. In *Int. Workshop on Graph Grammars and Their Application to Computer Science*, Williamsburg, Virginia, November 1994.

[WSK97] C. Welsch, A. Schalk, and S. Kramer. Integrating forward and reverse object-oriented software engineering. In *Proc of the 19th Int. Conf. on Software Engineering (ICSE 19), Boston, MA, USA. ACM Press*, 1997.