MBD Research Activities at Vienna University of Technology

Markus Stumptner and Franz Wotawa[1]

1 Introduction

Over the past 10 years the Database and Expert-system Group of the Institut für Informationssysteme located at the TU Wien has done research in many subfields of AI, including Model-based Reasoning (MBR). Within MBR algorithms for consistency-based diagnosis [7, 3], abductive diagnosis [4], and the integration of repair into the diagnosis process [5, 6, 8] has been explored.

In the last years research has focused on applying model-based diagnosis to software debugging [1]. This line of research was initiated by the DDV project funded by Siemens Austria. Section 2 gives an overview of the project and future research activities.

Another direction of research includes the object-oriented design of diagnosis kernels, providing a language for defining new diagnosis components and complete diagnosis systems (see section 3). Motivated by this work, our group has spent effort on building new more efficient diagnosis algorithms which can be used also for very large systems (up to 10.000 components and more). See [17] for an algorithm for tree-structured diagnosis systems.

Currently we are involved in a project with the aim of building scale-able diagnosis agents running on workstations, PCs, and personal digital assistants (PDAs), and are able to communicate diagnosis informations to each other. Section 4 gives an overview of this project.

2 Automatic Software Debugging

The basic principles of applying model-based diagnosis for software debugging is shown in figure 1 (a). The behavior of a program, given by the semantics of the used language, is compared with the intended behavior which is given by an oracle, i.e., the user or a (formal) specification. In case of behavioral differences discrepancies occur which are related to diagnosis candidates. In this context the piece of software can be viewed as system description and the oracle as physical system, which is contrary to the traditional model-based diagnosis view. There the system description is assumed to be correct and the physical system may contain a bug.

In figure 1 (b) the practical realization is depicted. The system description is composed out of a part, where the semantics of the programming language is modelled, and a part giving the structural information, i.e., used components and their interconnections, which can be automatically derived for every program using a compiler. From the specification based test cases we obtain the observations, e.g., variable values. The (logical) system description together with the observations determine the diagnosis candidates. Since program parts, e.g., expressions, statements, processes, procedures, ..., are directly mapped to diagnosis components, the computed diagnoses map back to the possible bug locations within the program.

For a discussion about other (automatic) software debugging approaches [2]. The application of model-based reasoning for program debugging and other intelligent debugging techniques is described in [19].

Since 1995, Siemens Austria has been funding the *DDV: 'Design Debugging of VHDL Specification'* project, aimed at improving the

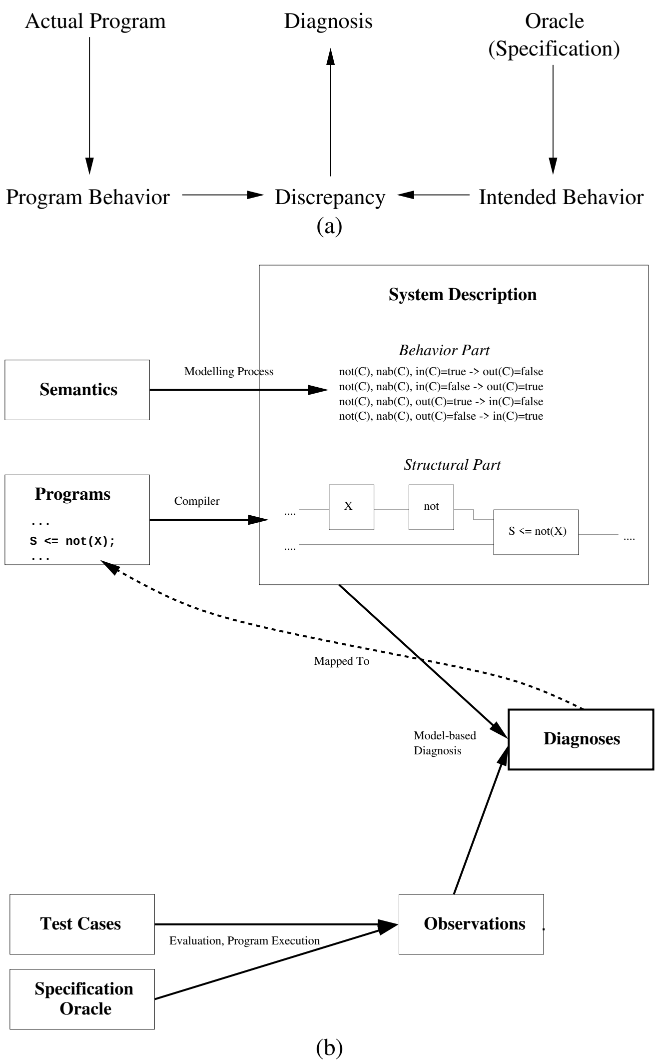

Figure 1. Principles of model-based software debugging

[1] Author's address: Technische Universität Wien, Institut für Informationssysteme, Paniglgasse 16, A-1140 Wien, Austria, Email: {mst,wotawa}@dbai.tuwien.ac.at

Submitted to ECAI 98
June 16, 1998

hardware design cycle. In hardware design the hardware description language VHDL [21] is used which is similar to *Ada*. VHDL programs have a concurrent part consisting out of processes using signals for communication. The behavior of processes has to be specified using sequential statements. VHDL programs are compiled and simulated, resulting in waveform traces, i.e., signal changes over time. Waveform traces (see figure 2 for a typical trace) are compared with a specification or another trace. In case of discrepancies the VHDL program contains a bug which has to be fixed.

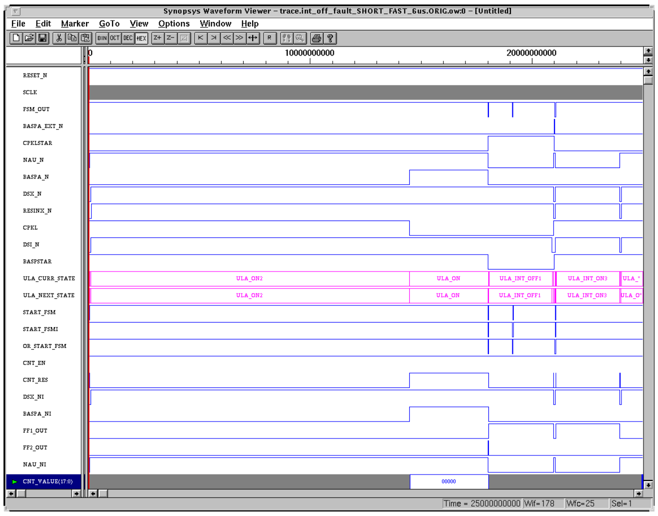

Figure 2. A typical waveform trace

In order to reach the project's aim two tools have been developed. These tools are fitted to the hardware designer's development environment, which consists out of a (vendor specific) compiler, simulator and a waveform trace viewer. The waveform compare tool (WFCOMP) automates the compare task and provides several different compare modes [16]. WFCOMP results can be directly used for locating the bug(s) using the debugging tool VHDLDIAG. Because of the program size (up to 10 MB source code and up to several days of simulation time for large designs) a very abstract model has been introduced for debugging [9, 16] which is suitable for focusing on fault candidates even for large programs. The model based on the dependencies between signals and variables given by the program structure which is similar (but not identical) to program slicing ([22, 23]).

Our group is currently working on improving the diagnosis capability, although these methods are initially intended to be only used with small VHDL programs. In particular, we have examined extensions of models for the description of VHDL programs based on modelling the correct behavior of

- concurrent signal assignments [15]
- the VHDL sequential part [14]
- the VHDL sequential statements considering semantical differences between signal and variable assignments together with the temporal consequences and natural diagnoses ordering criteria.[20]

Due to their rare occurrence in real-world VHDL designs, especially for synthesise-able programs, which do not allow their use at all, these models do not handle loop constructs and recursive functions. The behavior model described in [20] has been implemented and is currently tested using examples from real-world VHDL design projects.

Based on experiences from the DDV project, software diagnosis has been analysed on a more theoretical basis [13]. Results from previous research will be used in the project JADE 'Java Diagnosis Engine'. JADE is funded by the Austrian Science Fund (FWF) and has the aim of extending a Java debugger with automatic fault finding and repairing capability. JADE will begin in June 1998. One objective is to experiment with extensions to the proposed debugging approach to handle recursive functions, global variables, and other advanced constructs.

3 Diagnosis Engines

The objective of project *DIKE ('Diagnosis Kernel')*, a project funded by Siemens Austria, was to provide an object-oriented diagnosis framework. Generality, extensibility and providing methods for easy building and updating of diagnosis systems were the main requirements. The framework was implemented using ObjectShare Smalltalk Visualworks 2.52.

Within DIKE, diagnosis systems with hierarchical components and several behavioral modes can be described. Probabilities for all modes, repair costs and informations and observation costs can be specified. It has also been shown that the provided language, used for writing behavior descriptions and diagnosis systems, can be used in a wide range of application domains.

A number of example models were developed as part of the project. These include simple models for

- digital and analog circuits,
- reconfiguration of telecommunication systems [18],
- sound-switching systems used in radio and television broadcasting [12], a domain already examined previously in the context of other methods

The diagnosis engine, based on [11], has the capability of searching faults in a breadth-first order, depending on the number of components invoked or depending on the probability of a diagnosis. The maximum number of diagnoses and the maximum diagnosis size can be specified as parameters. In addition, the automatic generation of a fault mode and effect analysis (FMEA) is supported within the implemented framework.

4 Mobile Diagnosis Agents

The *MODIS: 'Mobile Diagnosis'* project funded by the *Hochschuljubiläumsstiftung der Stadt Wien* has the aim to build diagnosis agents with communication capability and which run on multiple platforms such as workstations, PCs, and PDAs. Currently, a language for describing the behavior and the structure of diagnosis systems is in development. This language will offer constructs for describing different behavioral modes, their probabilities, repair actions and repair costs, observation costs, physical impossibility axioms, among others. The description of hierarchical systems is also possible. The full syntax and semantics of the language together with a parser written in Smalltalk will be made available at the end of this year.

Other objectives of MODIS are:

- Defining an object-oriented framework for communication between multiple diagnosis and interface agents. In this part of the project, a communication language is used to transfer system descriptions, observations, results, and other diagnostic information. The framework itself must be flexible enough to provide communication via several channels, e.g., via email, remote file access, and socket-based communications.
- Improving diagnosis algorithms to be usable in PDAs for medium size diagnosis systems ($\approx$ 500 components). Since PDAs have restricted memory and computational power, algorithms must be optimised in order to reach the goal of providing a mobile diagnosis assistant with appealing user-interaction. Currently there are several algorithms available which can be used. Some are general (e.g., [10]) and others are restricted to systems satisfying certain structural properties (e.g., [17]).
- Defining a framework for the graphical user interface of diagnosis systems. Different resolution sizes (from 320$\times$240 black-and-white up to 1024$\times$800 with 16 Mio Colours) and input devices (mouse, keyboard, touch-screen) must be supported.

Figure 3 gives an overview of the MODIS objectives.

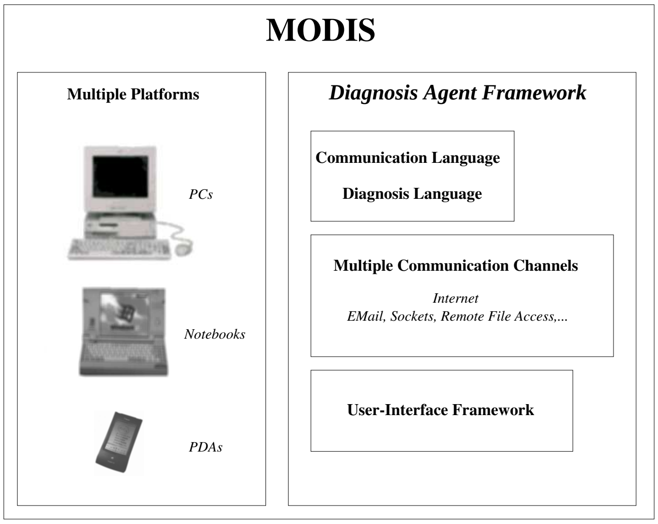

Figure 3. The MODIS project

5 Outlook on Future Activities

More recent projects include studies on the topic of monitoring technical systems, e.g., in facility management. Work in this area requires more sophisticated representation of temporal aspects.

Acknowledgement

This work has been supported in part by research grant DDV GR 21/96106/4 from Siemens Austria, by grant H-00031/97 from the Hochschuljubiläumsstiftung der Stadt Wien, and by the Christian Doppler Laboratory for Expert Systems.

REFERENCES

[1] Luca Console, Gerhard Friedrich, and Daniele Theseider Dupré, `Model-based diagnosis meets error diagnosis in logic programs', in *Proc. IJCAI*, pp. 1494–1499, Chambery, (August 1993). Morgan Kaufmann Publishers, Inc.

[2] Mireille Ducassé, `A pragmatic survey of automatic debugging', in *Proceedings of the 1st International Workshop on Automated and Algorithmic Debugging, AADEBUG '93*, Springer LNCS 749, pp. 1–15, (May 1993).

[3] Hartmut Freitag and Gerhard Friedrich, `Focusing on independent diagnosis problems', in *Proc. KR '92*, Cambridge, MA, (October 1992).

[4] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl, `Hypothesis classification, abductive diagnosis and therapy', in *Proceedings of the International Workshop on Expert Systems in Engineering*, Vienna, (September 1990). Springer Verlag, Lecture Notes in Artificial Intelligence, Vo. 462.

[5] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl, `Towards a theory of the repair process', in *Proceedings of the Portuguese Conference on Artificial Intelligence*, Albufeira, (October 1991). Springer AI Lecture Notes. Also appeared at the Model-Based Reasoning Workshop, AAAI'91, July 1991, Anaheim.

[6] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl, `Formalizing the repair process', in *Proc. ECAI*, Vienna, (August 1992). Appeared in Second International Workshop on Principles of Diagnosis, Milano September 1991.

[7] Gerhard Friedrich and Wolfgang Nejdl, `MOMO — Model-based diagnosis for everybody', in *Proceedings of the IEEE Conference on Artificial Intelligence Applications (CAIA)*, Santa Barbara, (March 1990). A slightly revised and extended version appears in Readings in Model-Based Diagnosis (Morgan Kaufmann, 1992).

[8] Gerhard Friedrich and Wolfgang Nejdl, `Choosing observations and actions in model-based diagnosis-repair systems', in *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pp. 489–498, Cambridge, MA, (October 1992). Morgan Kaufmann Publishers, Inc.

[9] Gerhard Friedrich, Markus Stumptner, and Franz Wotawa, `Model-based diagnosis of hardware designs', in *Proc. ECAI*, Budapest, (August 1996).

[10] Peter Fröhlich and Wolfgang Nejdl, `A Static Model-Based Engine for Model-Based Reasoning', in *Proc.* 15^{th} *IJCAI*, Nagoya, Japan, (August 1997).

[11] Raymond Reiter, `A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).

[12] Johannes Retti, Wolfgang Nejdl, and Gerhard Friedrich, `ARTEX — Knowledge based fault detection and correction in a switching system', in *Proceedings of the International GI Conference on Knowledge Based Systems*, München, (1987). Springer-Verlag.

[13] Markus Stumptner and Franz Wotawa, `A model-based approach to software debugging', in *Proceedings on the Seventh International Workshop on Principles of Diagnosis*, Val Morin, Canada, (1996).

[14] Markus Stumptner and Franz Wotawa, `Model-based diagnosis of hardware description languages', in *Proc. Computational Engineering in Systems Applications CESA'96 IMACS Multiconference, Invited Session on Model-Based Systems*, Lille, (1996).

[15] Markus Stumptner and Franz Wotawa, `Model-based program debugging and repair', in *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Fukuoka, (1996).

[16] Markus Stumptner and Franz Wotawa, `A model-based tool for finding faults in hardware designs', in *Proceedings Artificial Intelligence in Design*, Stanford, (1996).

[17] Markus Stumptner and Franz Wotawa, `Diagnosing tree-structured systems', in *Proceedings* 15^{th} *International Joint Conf. on Artificial Intelligence*, Nagoya, Japan, (1997).

[18] Markus Stumptner and Franz Wotawa, `Model-based reconfiguration', in *Proceedings Artificial Intelligence in Design*, Lisbon, Portugal, (1998). To appear.

[19] Markus Stumptner and Franz Wotawa, `A Survey of Intelligent Debugging', *AI Communications*, **11**(1), (1998). To appear.

[20] Markus Stumptner and Franz Wotawa, `VHDLDIAG+:Value-level Diagnosis of VHDL Programs', in *Proceedings on the Ninth International Workshop on Principles of Diagnosis*, Cape Cod, (May 1998).

[21] IEEE Standard VHDL Language Reference Manual LRM Std 1076-1987, 1988.

[22] Mark Weiser, `Programmers use slices when debugging', *Communications of the ACM*, **25**(7), 446–452, (July 1982).

[23] Mark Weiser, `Program slicing', *IEEE Transactions on Software Engineering*, **10**(4), 352–357, (July 1984).