Project: P12344-INF

Java Diagnosis Experiments

Request for Prolongation

1 Introduction

Project P12344-INF was started in June 1998 with a scheduled runtime of two years. In this document we intend to describe the progress made in the project so far and propose a plan for prolongation of the project in view of the new insights gained during the research. In general, the project has been very successful with the original research goals being achieved or exceeded. This is documented by the follwing:

- We have produced an implementation of a model-based debugger prototype for an imperative language, a first in debugging and diagnosis research literature.
- The debugger prototype incorporates different formal models for modeling Java programs for debugging, with one model being used and one undergoing testing.
- The work is very much accepted by the model-based research community as exhibited by the publications and presentations at the main workshops and conferences in the field, including a paper at IJCAI 99.
- The project is following the projected schedule.
- The prototype is fully functional with regard to the covered subset of the Java language, complete with graphical user interface.

In terms of overall publication activity, goals have been exceeded with numerous workshop and conference papers (all refereed by multiple reviewers) published or accepted for publication and several more submitted. These include a paper in the main program of IJCAI '99 (the biennial conference generally acknowledged as the foremost in the field) and ECAI 2000 (the primary European conference on AI). The planned journal papers are currently under preparation.

2 Progress Report

In this section we present current progress made in the project, with the original project plan given for comparison purposes.

- Task **T1** involved the theoretical investigation of Model-Based Software Debugging (MBSD).
  - *a compilation of an overview of the existing literature on model-based diagnosis and debugging*

    This has reached about eight hundred entries by now.
  - *Developing a classification of programming errors, with input from the literature on programmer cognition: types and occurrences of errors, effects of different types of errors etc.*

    This turned out to be of relatively little importance to the representations because the models we currently use turned out experimentally to work a priori; the task continues as a result of insights gained from experimentation.
  - *Development of a representation for causality in programs*
  - *Development of a representation of imperative languages for diagnosis*

    These points were crucial to the development of both models in use in the current debugger prototype and were therefore examined in detail; a specification of the process and the resulting system descriptions can be found in [MSW99a, MSW99b, MSWW00a].
  - *Adaptation and development of MBD algorithms for MBSD*

    This was a crucial task that was a prerequisite for the implementation of the current prototype debugger.
- Task **T2** dealt with the application of MBSD principles specificalyl to object-oriented (Java) programs.
  - *Examination of representation of the semantics Java constructs in a diagnosis context*
  - *Examination of class boundaries as abstraction borders for diagnosis*

    Both these points had to be solved to enable the modeling of Java programs, particularly with a more detailed, value oriented model. They were a prerequisite for the detailed model that was developed in the project and include the work presented in [MSW00b, MSW00c, MSWWa, SW99a].

- *Modeling of the standard Java class library for MBSD*

  This task is still ongoing. However, due to experiences made during the project it was decided to slightly alter the focus of this point because the Java language is still undergoing fairly strong evolutionary change. To easen the onerous task of coding the models for the individual Java classes, it was decided to develop a higher level language for model description. Earlier versions of this language had been developed in previous projects. It was decided to improve the language to make it suitable for describing software diagnosis models [HSW00].

- Task **T3** involved the study of the use of alternative models for diagnosis. In this area, we studied the examination of a logic programming approach to formulating diagnosis models.

  In order to debug a program P, we firstly compile P into a logical rule-based model M. M is then given in the input of a diagnosis engine which localizes bugs in P (if any) and makes repair suggestions. The encoding of M is achieved by using the classical logic programming (LP). Hence, LP plays an important role in the framework of the JADE project. LP deals only with the boolean values *true* and *false*. However, the capability to manipulate quantitative values in LP would increase the performances of the JADE system. Consider for instance a variable v of P which is not directly accessible of the user, hence the user is not aware of the value of v. Thus, no observation regarding v can be given to the diagnosis engine. However, if the user has (rough) informations about the influence on v of other variables, and if LP was able to deal with quantitative values, observations like "variable v is correct with the probability p" could be used by the diagnosis engine, which would increase the flexibility of the system. A quantitative approach of LP could be also used to encode a realistic level of occurrence of common bugs which could be used for repair suggestions. A quantitative extension of LP able to encode the above properties was proposed in [Mat99a, Mat99b]. However, this branch of our research (as originally envisioned in the project plan) is completely theoretical at this point and not considered in the implementation.

- Task **I1** dealt with the development of an Intelligent Debugging Environment Assistant, basically the knowledge-based component of the Jade Debugger.

  - *Implementation of the Inference and generic MBD class library*
  - *Java representation specific classes*
  - *Java compiler classes*
  - *Testing compiler on example programs from* **I2**
  - *MBSD classes*

- *Testing and evaluation of diagnosis*
- *User interface development (implementation+testing)*

  All these subtasks have been successfully concluded.
- *Global system test*
- *User manuals*
- *Implementation of alternate models developed in* **T3**.

  This is ongoing work that will be finished by the end of the project.

• Task **I2** dealt with the compilation of a set of example programs

- *Definition of a catalog of examples*
- *Establishing an ordering of implementation tasks according to complexity and relevance of the examples*
- *Implementation of simple programs*
- *Implementation of complex example programs*

This was accomplished and the results are recorded in [Mat99c].

References

[HSW00] Thomas Havelka, Markus Stumptner, and Franz Wotawa. AD2L - a programming language for model-based systems (preliminary report). In *Proceedings of the Eleventh International Workshop on Principles of Diagnosis*, June 2000. To appear.

[Mat99a] Cristinel Mateis. Extending disjunctive logic programming by T-norms. In *Proceedings of the International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)*, 1999.

[Mat99b] Cristinel Mateis. Disjunctive logic programming extended by T-norms. In *AI Communications*, 2000. To appear.

[Mat99c] Cristinel Mateis. Jade – Java Test Cases. Technical Report DBAI-TR, Technische Universität Wien, Institut für Informationssysteme 184/2, Paniglgasse 16, A-1040 Wien, Austria, 1999.

[Mat99d] Cristinel Mateis. Jade – Subsets of Java considered for Debugging. Technical Report DBAI-TR, Technische Universität Wien, Institut für Informationssysteme 184/2, Paniglgasse 16, A-1040 Wien, Austria, 1999.

[MSW99a] Cristinel Mateis, Markus Stumptner, and Franz Wotawa. Debugging of Java programs using a model-based approach. In *Proc. DX'99 Workshop*, Loch Awe, Scotland, 1999.

[MSW99b] Cristinel Mateis, Markus Stumptner, and Franz Wotawa. A Functional Dependency Model for Java Programs. Technical Report DBAI-TR-99/xx, Technische Universität Wien, Institut für Informationssysteme 184/2, Paniglgasse 16, A-1040 Wien, Austria, 1999.

[MSW00a] Cristinel Mateis, Markus Stumptner, and Franz Wotawa. Locating bugs in Java programs – first results of the Java Diagnosis Experiments (Jade) project. In *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, New Orleans, 2000. Springer-Verlag. To appear.

[MSW00b] Cristinel Mateis, Markus Stumptner, and Franz Wotawa. A value-based diagnosis model for Java programs. In *Proceedings of the Eleventh International Workshop on Principles of Diagnosis*, June 2000. To appear.

[MSW00c] Cristinel Mateis, Markus Stumptner, and Franz Wotawa. Modeling Java programs for Debugging. In *Proceedings of European Conference on Artificial Intelligence (ECAI 2000)*, June 2000. To appear.

[MSWWa] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. Locating bugs in Java programs – first results of the Java diagnosis experiments (JADE) project. Submitted for publication.

[MSWWb] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. A model-based approach to debugging. Submitted for publication.

[MSWW00a] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. JADE – a step towards an intelligent debugger. In *Proceedings of the Eleventh International Workshop on Principles of Diagnosis*, June 2000. To appear.

[MSWW00b] Cristinel Mateis, Markus Stumptner, Dominik Wieland, and Franz Wotawa. JADE – AI support for debugging Java programs. In *3rd Workshop on Intelligent Software Engineering (WISE3)*, Limerick, June 2000. To appear.

[SW99a] Markus Stumptner and Franz Wotawa. Debugging Functional Programs. In *Proceedings* 16^{th} *International Joint Conf. on Artificial Intelligence (IJCAI 99)*, Stockholm, Sweden, August 1999.

[SW99c] Markus Stumptner and Franz Wotawa. Jade – java diagnosis experiments – status and outlook. In *IJCAI '99 Workshop on Qualitative and Model Based Reasoning for Complex Systems and their Control*, Stockholm, Sweden, 1999.

3 Plan of Research for 3rd Year

In the JADE project we have produced a solid foundation on which further research in the domain of debugging can rely. This includes the development of the current dependency and value-based models for an important subset of Java, as well as first empirical results which show improvements compared to traditional debugging. In addition, we have identified further research directions which could not be handled during the two year run of the JADE project. One direction is due to extensions of the model to handle the full Java semantics. This is necessary if the final goal is to provide a debugger that can be directly used in a production environment. Another point is coupling the debugger directly with standardized Java libraries, e.g., the JDK package, and with a Java runtime environment. Again, solving this problem is required for building a debugger that is to be used by programmers in a real-world environment. This latter capability would be a prerequisite for large-scale experimentation with the system, i.e., free use of the debugger prototype by programmers external to the project on actual applications. Basically these are incremental modifications based on the currently implemented prototype.

A third issue turned out to be a highly relevant research subject in its own right during the work on the project. This topic is the issue of multimodel diagnosis. One of the basic issues in model-based diagnosis is the development of good models and even in the subfield of software diagnosis the outcome of modeling can be quite different for different language paradigms, making different languages into interesting research topics of their own. However, the issue we became aware of during the project was that the interaction between different models of the same domain is also highly relevant, can significantly influence outcomes and has only seen limited exposure in the model-based literature, although its relevance has been recognized in the past.

The problem of multimodel diagnosis concerns the integrated use of multiple models for one and the same diagnosis case. While the original JADE project goals involved the development and comparison of different models for Java programs as exhibited by the dependency based and value based models, at the moment the selection between the two models happens under user control, and in addition the focus of the models (i.e., the decision which program parts to examine in more detail) is also handled interactively. Finally, at any one time during reasoning, the current prototype only uses one model to search for faults.

Multimodel reasoning can be used to produce higher flexibility along either of those axes.

- It enables the employment of models that are suitable for catching different types of errors, thereby providing for better identification. This would require the presence of a collection of different models, each capable of handling a different type of bug. This model library has to be developed for this purpose.

  For example, there should be a model which only returns statements that involve an access to a wrong variable. This model only returns (cardinality) minimal diagnoses if such a bug really causes the incorrect behavior of the program. In [BH93, BH95] the authors use exactly this assumption for debugging a large Assembler program, by combining static code analysis with a Bayesian network. However, their approach is not model-based, instead involving a combination of two separate components with hardcoded interaction principles.

  In contrast, a multimodel reasoner should at least in principle be able to utilize different models with one underlying reasoning mechanism (although the ability of model-based reasoning to select from a variety of mechanims is also an advantage and whould not be compromised by this), and might provide the ability to flexibly change between them as the situation requires.

- Using different models during reasoning also allows the incorporation of models with higher or lower abstraction levels. In particular, dependency-based models can be used to provide an initial focus on smaller areas of a large or very large software system, and more detailed models can then be used to identify the errors and error locations more closely. In other words, the combination provides good discrimination *and* performance. We have pointed out the potential of such a combination in past work, but so far no study of the possibility of actually combining the two methods under automatic control or choice of focus has been made.

- Finally, the use of multiple models has long been recognized as an important issue in unambiguously recognizing a particular class of errors, so-called structural errors [Böt95, Dav84]. In the component-based representation typically used by diagnosis systems, certain faults may result in the faulty circuit actually having a different structure, i.e., the connections between the components are altered compared to the original system. The classic example from the hardware domain is a so-called bridge fault, where a bridge on a circuit connects two lines and forces them to identical state even when those lines are logically independent and therefore not directly connected in the behavioral model typically used for diagnosis. A different model is needed to identify possible locations for such faults (e.g., showing physical layout in the circuit domain would point out potential bridge faults at points where lines run close to each other). This type of faults can occur easily in the software domain, e.g., by mistyping a variable name.

In general, assuming that it is possible to select a model on the basis of the given behavior, structure, and other external requirements, e.g., upper limits for diagnosis time, the use of multiple models makes eminent sense. We have engaged in earlier work concerning multiple models in the area of hardware design diagnosis, a topic which is highly relevant on its own due to the difference in the language paradigms and the different styles of thinking and development used by hardware engineers (who use VHDL) and software engineers (which use Java).

Using multiple models we expect a better discrimination of bug candidates, faster debugging, shorter diagnosis step runtimes, and more intuitive explanations for bug (similar to the hardware domain [Böt95]). The importance of multimodel reasoning is underlined by the fact that the EU *Network of Excellence in Model-based Diagnosis and Qualitative Reasoning (MONET)* is in the process of starting a subgroup for investigating the use of multiple models in diagnosis. This is a hot topic at the international level and is generally acknowledged as a highly relevant research topic by the international research community.

Other open questions of research in model-based debugging include the extension of the empirical analyses and development of more advanced models to provide a theory that makes it possible to write self-adaptive software. Although the present empirical analysis shows that model-based debugging can be used for locating and fixing a bug, we have so far only worked on small examples. (It should be noted that the same generally holds for the debugging research community.) Extended analysis should also consider large programs. On the other hand, self-adaptive software is a long-term research goat. Thinking about programs that can automatically detect and correct bugs and change their outcome based on user suggestions is very appealing and could have a significant impact on software engineering. However, providing such capabilities is clearly beyond the scope of the current project.

This list of open problems is not complete and contains only those current research issues most relevant for the topic of JADE . However, two of these issues problems which showed themselves as being particularly promising as well as relevant in terms of overall results during the JADE project, are the use of multiple models and more extensive empirical analysis than envisioned in the original proposal.

Because of the high relevance of multimodel research, its significance to software debugging, and the fact that such research as well as the ability to do more extensive experimentation builds directly on the current JADE *theoretical foundation and implemented environment, we request the prolongation of the* JADE *project for another year.*

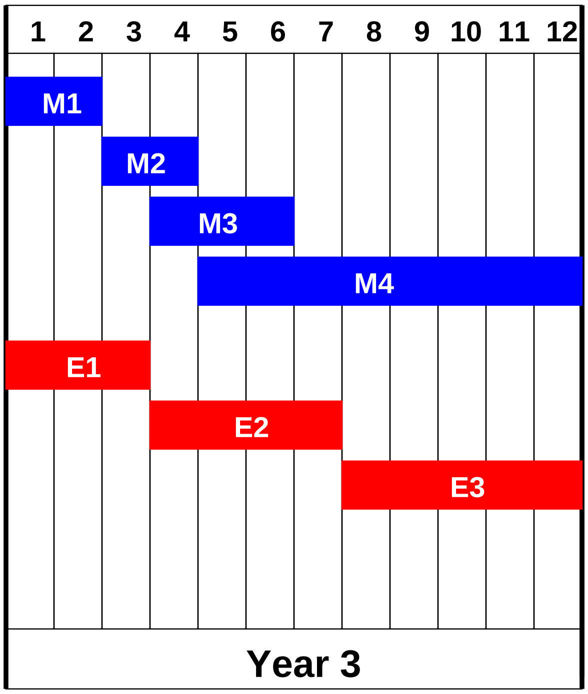
Multiple models Using multiple models required both the development of different logical models for programs, and a mechanism for reasoning about these models, with the aim of selecting the most appropriate model in a given situation. The final goal of this part of the project is the development of different models, their implementation using the available debugging framework, formalizing the model selection process, and its implementation. In order to reach this goal the following subtasks are required:

- **M1** Extending the value-based model to form a general model from which sub-models can be easily derived.
- **M2** Deriving two different models from the general value-based model handling:
  - structural bugs
  - functional bugs
- **M3** Formalize the model selection process. Identify important parameters allowing to select the most probable model.
- **M4** Implement the models and model selection using our debugging framework developed during the first 2 years of JADE .

Empirical analysis The empirical analysis is for showing improvements in debugging when using multiple models, comparing the effectiveness of the use of different models for different programs and error types, and for identifying open problems. We actually expect the detection of particular "hard cases" for the current models will lead to the development of new models of programs and hence foster research in this area. In summary, the following subtasks are planned in the empirical analysis category:

- **E1** Creating more test-cases, including the development of large Java applications/programs for this purpose. A detailed test plan has to be worked out. This plan must include the type of the model, the test cases to be used, and the expected output.
- **E2** Analyzing each model on its own using the developed test-cases. This analysis should show that the models are specialized for a particular kind of bugs and fail to deliver good answers for other bugs.
- **E3** Analyzing the model selection process. This analysis should show how the use of multiple models improves the overall debugging process.

The following schedule shows the subtask over time for a possible 3rd year of JADE . Most of the task, especially the development of the models and the construction of the test cases are to be accomplished by both research assistants.



4 Costs

We ask to extend the funding for the current personnel for another year. This would imply the following budget:

Project Member	Type	Duration	work extent	Costs
Dipl.-Ing. Dominik Wieland	Doct.stud.	12 months	100%	334.500,-
Dr. Cristinel Mateis	Postdoc	12 months	100%	554.000,-
Total				888.500,-

There are no requests for additional equipment will be used or costs arising from publications. No national or international cooperations within the JADE project are planned. No work on demand of third parties will be done within JADE .

5 Start Date

July 1, 2000.

6 Research location

Department of Databases and Artificial Intelligence
Institut of Information Systems
TU Vienna

7 Personnel available

- Ao.Univ.Prof.Dr. Markus Stumptner[1],
  Department of Databases and Artificial Intelligence
  Institut of Information Systems
  TU Vienna

- Univ.Ass. Dr. Franz Wotawa
  Department of Databases and Artificial Intelligence
  Institut of Information Systems
  TU Vienna

- Vertr.Ass. Dipl-Ing. Thomas Havelka
  Department of Databases and Artificial Intelligence
  Institut of Information Systems
  TU Vienna

References

[BH93] Lisa Burnell and Eric Horvitz. A Synthesis of Logical and Probabilistic Reasoning for Program Understanding and Debugging. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, pages 285 –291, 1993.

[BH95] Lisa Burnell and Eric Horvitz. Structure and Chance: Melding Logic and Probability for Software Debugging. *Communications of the ACM*, pages 31 – 41, 1995.

[Böt95] Claudia Böttcher. No faults in structure? How to diagnose hidden interaction. In *Proc. IJCAI*, Montreal, August 1995.

[Dav84] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[1]Currently on leave holding a full professorship at University of Linz until June 30, 2000.