Exploiting Models in Software Engineering

James Kiper, *Member, IEEE Computer Society,* Tim Menzies, *Member, IEEE Computer Society,* and Jeremy Greenwald,

***Abstract*— The use of models is a methodology that has proven its utility and effectiveness in many aspects of software engineering. Models in requirements engineering allow software developers to explore trade spaces more effectively. In software designs, software engineers have modeled static and dynamic structure of a system. Modeling checking, as with, for example., Spin, has greatly extended the ability of developers to verify various properties of software systems.**

However, building a useful model can be intellectually difficult. It often requires the cooperation of users, clients, and software engineers who speak various technical languages. Creating and validating such a model is a time-consuming, albeit important, process. Given the cost of model building, it is vital that each model be exploited as fully as possible.

In this paper, we illustrate how clear, succinct conclusions can be quickly extracted from a model by adopting a particular style of inference – minimal contrast set learning.

***Index Terms*— model-driven software engineering, contrast set learning**

I. Introduction

THIS demo file is intended to serve as a "starter file" for IEEE journal papers produced under LaTeX using IEEEtran.cls version 1.6b and later. [1] and [2] May all your publication endeavors be successful.

mds
November 18, 2002

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Modeling in Software Engineering

Software engineering is modeling. At each phase of the software life cycle, we produce artifacts that are, in fact, models. To be explicit, by the term *model*, we mean "elements describing something (for example, a system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis, such as

- Communication of ideas between people and machines
- Completeness checking
- Race condition analysis
- Test case generation
- Viability in terms of indicators such as cost and estimation
- Standards
- Transformation into an implementation"

[3]

Contact information: kiperjd@muohio.edu, timm@cs.pdx.edu, jegreen@cs.pdx.edu .

III. A Saturation Effect

IV. Related Work: Saturation Elsewhere

V. Contrast Set Learning

A. Contrast Set Learning with Tar2

VI. Case Studies

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] A. Avritzer and E. J. Weyuker, "The automatic generation of load test suites and the assessment of the resulting software," *IEEE Trans. on Software Engineering*, vol. 21, no. 9, pp. 705–716, September 1995.

[2] T. Menzies and Y. Huf, "Data mining for very busy people," *IEEE Computer*, vol. 36, no. 11, pp. 22–29, November 2003, available from http://menzies.us/pdf/03tar2.pdf.

[3] S. Mellor, A. Clark, and T. Futagamii, "Model-driven development - guest editor's introduction," *IEEE Software*, vol. 20, no. 5, pp. 14– 18, Sept.-Oct. 2003.

PLACE PHOTO HERE

James Kiper James Kiper is Associate Dean for Research and Graduate Studies in the School of Engineering and Applied Science, and Professor of Computer Science at Miami University where he has been for the past twenty years. His research interest are in the area of software engineering...

PLACE PHOTO HERE

Tim Menzies Tim Menzies is an Associate Professor in the Lane Department of Computer Science at West Virginia University. He is also ...

Jeremy Greenwald Jeremy Greenwald is a graduate student in the Department of Computer Science at Portland State University.