Benefits of Software Measures for Evolutionary White-Box Testing

Frank Lammermann
Daimler Chrysler AG, Research and Technology
Alt-Moabit 96a,
10559 Berlin, Germany
+49 - 30 - 399 82 272
Frank.Lammermann@DaimlerChrysler.com

Stefan Wappler
Daimler Chrysler AG, Research and Technology
Alt-Moabit 96a,
10559 Berlin, Germany
+49 - 30 - 399 82 358
Stefan.Wappler@DaimlerChrysler.com

ABSTRACT
White-box testing is an important method for the early detection of errors during software development. In this process test case generation plays a crucial role, defining appropriate and error-sensitive test data. The evolutionary white-box testing is a promising approach for the complete automation of structure-oriented test case generation. Here, test case generation can be completely automated with the help of evolutionary algorithms. However, problem cases exist in which the evolutionary test is not able to find valid test data. Thus, in the case of not achieving a test goal, it is not known whether this is due to non-executable program code or a problem case. This paper will investigate how successfully a software measure can support an evolutionary white-box test if the measure can predict the test effort. Hence, the termination criteria of evolutionary white-box testing can be adapted to test goals with problem cases in such a way that problematic test goals are either excluded from the test in advance or can be covered due to an adequate termination criteria according to a software measure. This could lead to an increase in efficiency and effectiveness of evolutionary white-box testing.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *Complexity Measures, Performance Measures.*

General Terms
Measurement, Verification

Keywords
evolutionary testing, software measurement, metric, automated test case generation

1. INTRODUCTION
For the early detection of errors during software development, the white-box test, in addition to the function-oriented test, constitutes an important process. For this the generation of test data takes on a decisive role, since it determines the quality-relevant input values for the test. A successful approach to automated test data generation is the *evolutionary white-box test* [1, 2, 3]. It proves difficult, however, to define suitable termination criteria for an evolutionary white-box test, since, in the case of non-achievement through the test-criterion-required program structure, it is not known if the program structure is not executable or whether test data was not searched for thoroughly enough.

This paper will experimentally examine the potential benefits of a software measure that estimates the test effort of an evolutionary white-box test. This so-called *evolutionary software measure* could make it possible to individually adapt the termination criteria to each test goal so that the test goal can generally be either reached or, if it is expected that it will not be reached, excluded before the test.

The experiment results in this chapter relate to the framework for evolutionary white-box testing[1], developed by DaimlerChrysler and described in [3]. For the experiments 23 test object were selected – individual C functions which are typical for the use of evolutionary white-box tests. They originate from different application areas such as mathematical calculations for control problems, the performance of string and character operations often found in programming language libraries, and components from automotive and motor electronics.

2. ADVANTAGES OF EVOLUTIONARY SOFTWARE MEASURES
The quality of objective functions plays an important role in determining the success of evolutionary white-box tests [1, 2]. Searching out valid test data, especially for complex test objects, can present difficulties if the objective function can not make details available for the optimization of test data. Such situations are designated in the following text as *non-achievability problems*.

For the *e*volutionary white-box test there exist non-achievability problems for which an evolutionary software measure can be

[1] At the moment only C programs can be tested with the framework.

useful, e.g. the problem of suitable parameterization, the problem of a suitable termination criterion, or the problem of a too-small optimum size. There exist also problems where such measures do not offer benefits, e.g. the flag problem [4], the problem of floating point variables or the state problem [5].

2.1 Frequency of non-achievability problems

In order to judge the efficiency of an evolutionary software measure, we consider the frequency of the occurrence of individual non-achievability problems using the 23 examined test objects. From the 767 total test goals, the evolutionary white-box test could not attain 181 test goals in at least one of five test runs due to the problems listed above. The frequencies of individual non-achievability problems are listed in table 1.

Table 1. Measured frequency of non-achievability problems

Problem Category	Problem Case	Frequency	Portion
with benefit through evolutionary software measure	Parameterization	?	?
	Termination criteria	54 from 181	29.8%
	Optimum size	32 from 181	17.7%
	Consecutive fault	77 from 181	42.5%
without benefit through evolutionary software measure	Flag problem	0 from 181	0.0%
	Floating point variable	4 from 181	2.2%
	State Problem	14 from 181	7.7%

The largest portion of the non-achievability problems is caused by the consecutive fault problem. The causes of the problem of consecutive fault are based, in turn, on other non-achievability problems. Of the 77 test goals that could not be attained due to consecutive fault, 71.4% were shown by further analyses to be caused by the problem of optimum size, 18.2% by the state problem, and 10.4% by the problem of floating point variables. If one considers those problems caused by consecutive fault when assessing the frequency of all the non-achievability problems, the total number of non-achievability problems for which an evolutionary software measure can offer some benefit climbs to 77.9%.

3. CONCLUSION

Our experiments have shown that the bulk of situations in software programs (77.9%), in which evolutionary white-box testing so far has not been able to make assertions about the existence of non-achievable test goals, can be improved with the help of an evolutionary software measure. This improvement is based on the ability of the evolutionary software measure to estimate in advance the test effort for single test goals of evolutionary white-box testing.

In order to develop such a software measure, those attributes in test objects have to be examined which are responsible for the amount of test effort. All of these attributes are restricted to characteristics of conditions, which have an influence on the structure of the search space.

Another step of research remains: how will the evolutionary software measure behave when evolutionary white-box testing is parameterized differently? Depending on the settings, it is possible that merely the constants given for the test effort calculation have to be adjusted. Furthermore, it is necessary to check how well the evolutionary software measure works within an environment of interfering influences (e.g. changes of input variables in dependency on conditions) or whether it is only working well in an artificial surrounding. In order to apply software measures for the calculation of a dynamic termination criterion, it is also necessary to examine the distribution of real measured test effort possesses in comparison to forecasted efforts by the software measure.

Acknowledgements

The work described has been performed within the SysTest project. The SysTest project is funded by the European Community under the 5th Framework Programme (GROWTH), project reference G1RD-CT-2002-00683.

4. REFERENCES

[1] Sthamer, H. *The Automatic Generation of Software Test Data Using Genetic Algorithms*. PhD Thesis, University of Glamorgan, Pontyprid, Wales, Great Britain, 1996.

[2] Baresel, A., Sthamer, H., and Schmidt, M. *Fitness Function Design to improve Evolutionary Structural Testing*. Proceedings of GECCO 2002, New York, 2002, 1329-1336.

[3] Wegener, J., Baresel, A., and Sthamer, H. *Evolutionary Test Environment for Automatic Structural Testing*. Information and Software Technology, vol. 43, 2001, 841-854.

[4] Harman, M., Hu, L., Hierons, R., Baresel, A., and Sthamer, H. *Improving Evolutionary Testing by Flag Removal*. Proceedings of GECCO 2002, New York, 2002.

[5] Harman, M., Hu, L., Hierons, R., Munro, M., Zhang, X., Dolado, J., Otero, M., and Wegener, J. *A Post-Placement Side-Effect Removal Algorithm*. Proceedings der IEEE International Conference, New York, 2002.