Towards Research on Software Cybernetics *

Kai-Yuan Cai
Department of Automatic Control
Beijing University of Aeronautics and Astronautics
Beijing 100083, China
Email: kyc@ns.dept3.buaa.edu.cn

T. Y. Chen
School of Information Technology
Swinburne University of Technology
Hawthorn 3122, Australia
Email: tychen@it.swin.edu.au

T. H. Tse †
Department of Computer Science and Information Systems
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: tse@csis.hku.hk

Abstract

Software cybernetics is a newly proposed area in software engineering. It makes better use of the interplay between control theory/engineering and software engineering. In this paper, we look into the research potentials of this emerging area.

Keywords: Control engineering, software engineering, software cybernetics

1. Introduction

Software cybernetics is an emerging area in software engineering that explores the interplay between control theory/engineering and software engineering [1].

As a matter of fact, the interplay between software and control is not new. In computer-controlled systems, control policies or algorithms must be implemented in embedded software. There are various software tools, such as Matlab [2], which support control systems design. Theoretical software models may also help to develop control theories. Finite-state automata, for instance, have been used to represent discrete event dynamic systems, leading to a supervisory control theory in the control community [3, 4].

The existing interplay between software and control is, however, far from comprehensive. For example, control theories seldom play a major role in the quality assurance of software analysis and design. Software theories, such as temporal logic, CSP and CCS, have not been fully applied in control systems analysis or synthesis either. The core of software engineering and that of control theory/engineering have been developed independently of each other. The feasibility of bringing these two cores together has rarely been explored.

Our proposed concept of software cybernetics is to make better use of the interplay between the core of software engineering and that of control theory/engineering. It treats software problems as a control problem, and control problems as a software problem. It deals with software problems and control problems in an integrated manner.

2. Software Testing in the Context of Software Cybernetics

The feasibility and effectiveness of software cybernetics have been demonstrated in the area of software testing by treating the latter as a control problem [1]. The software under test serves as the controlled object, while the software testing strategy serves as the corresponding controller. The software under test and the corresponding testing strategy make up a closed-loop feedback control system. In this way, the feedback mechanism in software testing is formalized.

By treating software testing as a control problem, we can address an inverse problem of software testing: Given a quantitative testing or reliability goal, how can we design an optimal testing strategy to achieve this goal?

We should explore adaptive strategies in software testing. Adaptive testing is the counterpart of adaptive control in software engineering. It means that a software testing strategy should be adjusted online as a consequence

*This research is supported in part by the Research Grants Council of Hong Kong and the University of Hong Kong Committee on Research and Conference Grants.

†Contact author.

1530-2059/02 $17.00 © 2002 IEEE

Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)
1530-2059/02 $17.00 © 2002 IEEE

of the test results collected during software testing, since the understanding of the software under test is improved. A non-adaptive software testing strategy specifies the whole test suite to be generated. On the other hand, an adaptive software testing strategy specifies the next testing policy to be employed for a given history of testing events. This new testing policy in turn determines the next test cases to be generated. In comparison with random testing (which is commonly adopted as a benchmark for softwate testing), adaptive testing uses fewer tests to detect more software defects. Some initial but encouraging results have been obtained [1, 5].

3. Other Directions in Software Cybernetics

Besides the area of software testing, we propose the following directions in software cybernetics. Some preliminary studies have been made in [6].

(*a*) *Software Control Policies with Different Tolerances for Failures*

Modern aircraft adopt fly-by-wire flight control systems. Flight control laws are implemented with embedded software components and systems. Similarly, different software control policies may have different degrees of tolerance towards software failures. In order to obtain satisfactory software control policies, both control theory and software reliability theory should be taken into account. Neither of them can be overlooked.

(*b*) *Software Testability, Software Controllability and Software Observability*

Software testability refers to the ability that software defects can be detected. Software controllability refers to whether a testing process can be generated. Software observability refers to whether a testing process can be recorded. On the other hand, the controllability of a control system often refers to whether the desired state transfer can be implemented, and the observability of a control system often refers to whether an initial state can be estimated using the finite history of input-output pairs of the system. Can these concepts be unified in a single framework with a view to improving the software quality?

(*c*) *Bisimulation and Controllability*

Bisimulation is a key notion in concurrent software processes, and controllability is a key notion in control systems. How can we define the controllability of concurrent software processes? Can a general and unified theory of bisimulation and controllability be developed?

(*d*) *Cybernetic Software Engineering*

Generalizing on the philosophy of treating software testing as a control problem, we may also treat software development as a control problem. This should lead to a new form of software engineering, potentially known as cybernetic software engineering. The software under development serves as a controlled object and the software development process serves as the corresponding controller. The software under development and the software development process, together, constitute a closed-loop feedback control system.

4. Comparison with Artificial Intelligence

Another interesting point is that artificial intelligence theories and control theories follow different philosophies, and hence their applications to software engineering problems may yield different results. In comparison with control theories, AI theories are more concerned with general-purpose techniques of problem solving. The underlying feedback mechanism does not play a central role. However, a control theory normally focuses on a class of controlled objects. The controlled objects and the corresponding controller must be distinguished from one another. The feedback mechanism from a controlled object to the corresponding controller plays a key role in synthesizing the controller to achieve a given control goal.

5. Conclusion

From the above overview, the software cybernetics should be a promising area for research and practice in the integration of software engineering and control engineering.

References

[1] K.-Y. Cai, "Optimal software testing and adaptive software testing in the context of software cybernetics", *Information and Software Technology* **44** (2002) (to appear).

[2] D. Hanselman and B. R. Littlefield, *Mastering Matlab 6*, Prentice Hall, Englewood Cliffs, New Jersey (2001).

[3] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems", *Proceedings of the IEEE* **77** (1): 81–98 (1989).

[4] K. C. Wong, J. G. Thistle, R. P. Malhame and H.-H. Hoang, "Supervisory control of distributed systems: conflict resolution", *Discrete Event Dynamic Systems: Theory and Applications* **10** (1/2): 131–186 (2000).

[5] K.-Y. Cai, T. Y. Chen, Y.-C. Li and Y. T. Yu, "On the on-line parameter estimation problem in adaptive software testing" (in preparation).

[6] C. Bai, K.-Y. Cai and T. Y. Chen, "Necessary/sufficient conditions for software defect curves" (in preparation).

Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)
1530-2059/02 $17.00 © 2002 **IEEE**