How to Fake a Rational Design Process using the SCR Method

In *Proc. Software Engineering for High Assurance Systems (SEHAS 2003)*, **held in conjunction with ICSE 2003, May 9–10 2003. Portland, Oregon, USA.**

Ramesh Bharadwaj
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington DC 20375 USA
ramesh@itd.nrl.navy.mil

Abstract

We explore the idea of faking a rational design process, a la Parnas and Clements [7], by the application of the extended SCR Method of Heitmeyer and Bharadwaj. We argue that the formal artefacts created as a result serve as the basis for determining the work products associated with each step of the process, and whose quality assessment is aided by the application of tools in the SCR Toolset. Further, since the products associated with each step have a consistent formal denotation, the approach opens the possibility of significantly automating many process steps.

1. Introduction

The term “Rational Design Process” was introduced by Parnas and Clements in [7] to define an ideal process in which programs are derived from the requirements in the same way that theorems are derived from axioms in a proof. However, they argue, it is impossible to find a process in practice that designs software in a perfectly rational way, and therefore suggest that we fake such a process, i.e., we present our system and its associated documents to others as if we had followed this idealized design process. They justify the need for this pretense by arguing that: (a) An understanding of the ideal process provides designers guidance on how to proceed. (b) People will get closer to a rational design if they try to follow the process. (c) It is a reasonable basis for specifying a “standard process”. (d) It is easier to provide metrics on a project if it is compared with the ideal process. (e) Attempting to follow a standard process eases a project’s review. Finally, they argue that management of such a process that is described in terms of work products becomes easier since we know which work products are due, and what criteria they must satisfy.

In [1, 3] Heitmeyer and Bharadwaj outline a four step process for the behavioral specification of software-intensive embedded systems. Each step in the process results in the construction of a formal artefact that is amenable to formal inspection, validation, verification, and consistency checking. In this paper, our tentative proposal is that we additionally consider two architectural specification documents. Although these artefacts may differ somewhat from the products considered by Parnas and Clements in [7], we argue that the SCR notation, the SCR formal model [5], and tools associated with the SCR method [2, 4] can serve as the basis for the construction of the products of a rational design process. The benefit of this approach is that many of the desirable properties of the products considered in [7] are natural outcomes of the application of the SCR method and tools. We argue therefore that adopting the SCR method and using the SCR tools leads us closer to the idealized design process because of the separation of concerns in each step of the process and precise formal criteria upon which the artefacts, constructed as a result of each step, are evaluated. What remains to be done is to provide a set of guidelines suitable for use by developers and precise documentation of the “SCR Rational Process” itself, along the lines suggested in [7].

2 Behavioral Specifications

Behavioral specifications of the system and software are constructed using the notation of SCR. The four steps of the extended SCR method of [1, 3] are shown in Figure 1.

The first step creates the *System Requirements Specification* (SRS), which describes the required external behavior of the system in terms of the quantities in the environment that the system monitors and controls. The structure of the SRS is dictated by the structure of the system architecture (see next section) – the SRS may be composed of modules, each corresponding to an element in the system architectural specification. The second step creates the *System Design Specification* (SDS), which identifies the input and output variables associated with the sensors and actuators on each hardware platform of the system. The third step creates the *Software Requirements Specification* (SoRS) by ex-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2003**	2. REPORT TYPE	3. DATES COVERED **00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE **How to Fake a Rational Design Process using the SCR Method**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**2**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

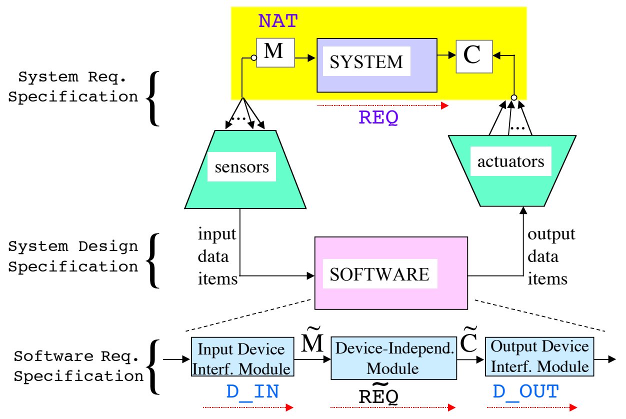

Figure 1. Relationship between the artefacts of the SCR method.

tending each SRS module with two additional specifications that (1) relates the input devices to the monitored quantities of the system and (2) the controlled quantities of the system to the output devices. The fourth step extends the SRS with behavior to handle hardware malfunctions such as sensor and actuator failures, and timing and accuracy constraints.

3 Architectural Specifications

In this paper, we propose that each behavioral specification should be accompanied by an associated architectural specification. The notation we propose for architectural specifications is similar to wiring diagrams used by Electrical Engineers to specify component interconnections. The *System Architectural Specification* documents the proposed physical configuration of the computer hardware. The *Software Architectural Specification* documents the internal structure of the software on each computing platform. If the underlying software is designed using an object-oriented method, then a class diagram should probably be included as an adjunct to the software architectural specification. The class diagram serves as a surrogate for documenting the module structure, module interfaces, and the uses hierarchy as proposed in [7].

4 Software Design and Implementation

The software design document records the major design decisions of each module (or class) which is intended to allow a quick review of the design before coding begins [7]. This includes descriptions of the internal data structures and the major algorithms used in the implementation. It is important to note that, with certain restrictions, it is possible to generate efficient and verified code directly from the software requirements specification. This has remained a search for the philosopher's stone with other specification notations such as UML. However, with SCR specifications, it is possible to automate substantial portions of the code creation process as demonstrated by Heitmeyer et al. [6].

5 Conclusions

We demonstrate that the SCR method is well suited for faking the ideal development process. In addition to conventional reviews and inspection, specifications in SCR are amenable to mechanical checking and verification, thereby considerably easing the burden on the designers and reviewers. Adopting SCR also promotes seamless transition between the process steps, since all the artefacts have a consistent underlying formal semantics, based on the SCR formal model[5]. Finally, the behavioral descriptions in SCR have a precise, mathematical semantics and therefore lend themselves well to the automatic generation of substantial portions of the implementation. Parnas and Clements say in conclusion in [7] that being a rational designer is very hard, and that even faking that process can be difficult. We posit that, for the designer, adopting the SCR method and using the SCR tools will considerably ease this process.

References

[1] R. Bharadwaj and C. Heitmeyer. Hardware/software co-design and co-validation using the SCR method. In *Proceedings of the IEEE International High Level Design Validation and Test Workshop (HLDVT'99)*, San Diego, CA, Nov. 1999.

[2] R. Bharadwaj and S. Sims. Salsa: Combining constraint solvers with BDDs for automatic invariant checking. In *Proc. 6^{th} International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), ETAPS 2000*, Berlin, Mar. 2000.

[3] C. Heitmeyer and R. Bharadwaj. Applying the SCR requirements method to the Light Control Case Study. *Journal of Universal Computer Science*, 6(7), 2000.

[4] C. Heitmeyer, J. Kirby, Jr., B. Labaw, and R. Bharadwaj. SCR*: A toolset for specifying and analyzing software requirements. In *Proc. Computer-Aided Verifi cation, 10th Annual Conf. (CAV'98)*, Vancouver, Canada, 1998.

[5] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw. Automated consistency checking of requirements specifi cations. *ACM Transactions on Software Engineering and Methodology*, 5(3):231–261, April–June 1996.

[6] E. I. Leonard and C. L. Heitmeyer. Program synthesis from formal requirements specifi cations using APTS. *Higher-Order and Symbolic Computation*, To Appear.

[7] D. L. Parnas and P. C. Clements. A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng.*, SE-12(2):251–257, Feb. 1986.