Science and Engineering: a collusion of cultures

Tony Hoare
Microsoft Research Ltd., Cambridge
thoare@microsoft.com

Abstract

The cultures of science and engineering are diametrically opposed along a number of dimensions: long-term/short-term, idealism/ compromise, formality/ intuition, certainty/risk management, perfection/ adequacy, originality/familiarity, generality/specificity, unification/diversity, separation/amalgamation of concerns. You would expect two such radically different cultures to collide. Yet all the technological advances of the modern era result not from their collision but from their collusion – in its original sense of a fruitful interplay of ideas from both cultures.

I will illustrate these points by the example of research into program verification and research into dependability of systems. The first of these aims at development and exploitation of a grand unified theory of programming, and therefore shares more the culture of science. The second is based on practical experience of projects in a range of important computer applications, and it shares more the culture of engineering. A collision of cultures would not be unexpected. But I will suggest that the time has come for collusion, and I will suggest how. We need to define an interface across which the cultures can explicitly collaborate.

Dependability research can deliver its results in the form of a library of realistic domain models for a variety of important and common computer applications. A domain model is a reusable pattern for many subsequently conceived products or product lines. It includes a mix of informal and formal descriptions of the environment in which the computer system or network is embedded. It concentrates on the interfaces to the computer system, and the likely requirements and preferences of its community of users. The practicing software engineer takes the relevant application domain model as the starting point for a new project or project proposal, and then specializes it to accord with the current environment and current customer requirements. Domain models are most likely to emerge as the deliverable result of good research into dependability.

If the available tools are powerful enough, verification can begin already at this stage to deliver benefit, by checking the consistency of formalized requirements, and detecting possible feature interactions. Ideally, implementation proceeds from then on in a manner that ensures correctness by construction. At all stages the project should be supported by verification tools. That is the long-term goal of a new initiative in Verified Software, which is under discussion by the international computing research community. This initiative has both a scientific strand and an engineering strand. The scientific strand develops the necessary unified and comprehensive theories of programming; it implements the tools that apply the theory to actual program verification; and it tests both the theory and the tools by application to a representative corpus of real or realistic programs. The engineering strand develops a library of domain models and specifications which enable practicing engineers to apply the theory and the tools to new programs in the relevant application domain. We hope that the results of this research will contribute to the reduction of the current significant costs of programming error. To achieve this will require a successful collusion of the scientific and engineering cultures.