Postprint of invited paper presented in the 47th Meeting of the IFIP Working Group WG 2.1
on Algorithmic Languages and Calculi (1995)

A Tale of Two Cultures:

Towards a Bridge between Software Mathematicians and Software Craftsmen*

T.H. Tse
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong

There are two distinct cultures in software engineering circles. The dominating culture, as shared by most practitioners, regards software engineering as a craft. Examples are structured analysis and design and object-oriented analysis and design. These methods recommend heuristic procedures based on the experience of prominent consultants in the industry. They are supported by user-friendly CASE tools with window-based graphics. Unfortunately, many of the systems thus developed do not live up to expectations.

The other culture proposes formal mathematically-based methods. These methods enable us to specify the requirements of target systems in an unambiguous manner. We can then verify whether the implemented systems behave according to the specifications using proofs of correctness. Very few formal methods have found their ways to the real world, however, since the mathematical jargon involved is generally not accepted by the industry.

We propose a bridge between the two cultures. We recommend that software engineers should not be craftsmen who simply follow the practices of some masters. Neither should they be mathematicians who concentrate only on fundamentals, without much regard to industrial practices. Real software ‘‘engineers’’ should be proficient in both theory and practice.

The popular graphical notations are useful for abstracting ideas and as blueprints to users. Their use should be continued in software engineering. At the same time, these blueprints must have a mathematical foundation, which is essential for reasoning and verification. The graphical and the mathematical forms should be convertible into each other as far as possible, so that software engineers can chose the appropriate representation of the target system according to their needs.

To illustrate our philosophy, we present a project entitled Functional Object-Oriented Design (FOOD). It provides a bridge between the popular graphical notations of Object Modelling Technique (OMT) and the algebraic notions of Functional Object-Oriented Programming System (FOOPS). The static relationships in a system, such as classes, methods, attributes and inheritance, can be specified using object diagrams and data flow diagrams, and is given a semantics of algebraic functions. The behavioural properties of the system, which can be specified by state diagrams and data flow diagrams, is given a semantics of algebraic equations.

---

* Part of this research was done at the Programming Research Group of the University of Oxford under an SERC Visiting Fellowship and an ACU Visiting Fellowship. The research is also supported in part by a Research and Conference Grant of the University of Hong Kong.