# Foreword

## Problem frames and software engineering

In welcoming this issue of *Expert Systems*, I would like to take the opportunity to remark on a relationship between the problem frames approach to software engineering and some practices of established engineering branches.

Common use of the phrase 'software engineering' dates from 1967, when a NATO Science Committee Study Group recommended the holding of a working conference by that name. The name was deliberately provocative 'in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering'. The proposed conference was held in 1968, and a follow-up conference in 1969.

In some respects software has made huge strides in the past 40 years, and software systems are available today that would have been scarcely thinkable in 1967. But in some other respects the original motivation of the NATO conferences has been neglected. Software development has learned remarkably little from the established branches of engineering. Indeed, their proffered lessons were ignored even in the two NATO conferences themselves: a scan of the conference reports shows that with one possible exception – Doug McIlroy's plea for standard, off-the-shelf software components – no substantial attention was paid either to their theoretical foundations or to their practical disciplines.

The problem frames approach to software development could be regarded as an attempt, both in original intention and in the path taken by subsequent work on the topic, to learn at least one lesson from the established engineers. This lesson is the importance of what Walter Vincenti calls *normal design*. An engineer practising normal design already knows the standard form of the device or product to be designed, how its parts fit together, and the operational principle by which it will achieve its purpose. By faithfully following normal practice the engineer can have a good degree of confidence that the result will be successful. In the software context, the standard subproblem classes proposed by the problem frames approach – workpieces, information display, required behaviour and so on – can be regarded similarly, as categories of objects of normal design in which well-established and well-understood precedents can govern both the form of the product and the development practices from which it emerges.

Failure is a matter to which established engineering branches pay a great deal of attention. Success comes from experience, it is said, and experience comes from failure. One vital aspect of normal design practice is a twofold approach to the avoidance of failure. First, the standard normal design itself embodies important lessons from past failures. Even when awareness of particular dramatic failures has faded in the collective consciousness of the design engineers, the design choices devised to avoid those failures remain part of the normal product design, and work to prevent a repetition. Second, normal design practice specifically includes knowledge of respects in which designs of the product class in question can still be liable to fail. For example, ever since the dramatic collapse of the Tacoma Narrows Bridge in 1940, designers of suspension bridges are very well aware of the danger of failure due to vertical oscillation of a shallow and narrow roadbed: they do not regard their design work as complete until this danger has been explicitly considered and excluded. In a similar fashion, the repertoires of subproblem

and composition concerns identified in the problem frames approach point up likely possibilities of failure, and demand to be explicitly addressed as an integral part of the design work.

The four papers in this journal issue relate problem frames to four very different technologies of software engineering. Each is worth careful reading in itself as an individual contribution. Further, their variety offers some confirmation that the ideas underlying the problem frames approach have a broad import, and augurs a fruitful future.

Michael Jackson
*The Open University*
*E-mail: jacksonma@acm.org*