

Position Paper: Codifying and Synthesising Software Design Knowledge

David Budgen
Department of Computer Science
Keele University
Staffordshire ST5 5BG
UK

Introduction

The act of designing something, in any domain, involves a complex and creative set of activities. When designing software, this is further compounded by the *invisibility* of the medium, and by the lack of readily visualisable representations of software artefacts.

Over the years, the software engineering community has evolved various ways of addressing many of these issues. For complexity we use a mix of hierarchical forms of notation and practices that seek to sub-divide the problem (separation of concerns). Creativity is usually structured by using design methods and patterns as vehicles for knowledge transfer, both between peers, as well as between teachers and students. Invisibility is normally addressed by employing a range of ‘box and line’ notations describing different ‘viewpoints’, such as construction, behaviour, function and data modelling (Budgen, 2003).

Useful as these undoubtedly are, a significant weakness is the lack of any over-arching evidence-based framework as the basis for their deployment. Design methods use procedures that are essentially distilled from the experiences of individuals, and notations constrained almost exclusively to ‘box and line’ forms. While this need not imply a lack of usefulness, such an informal basis for distillation of experience, together with the constraints imposed by the limitations of procedural forms, does not provide a particularly ‘scientific’ foundation. Design patterns are derived from a collective view of what is a ‘good’ solution form—and while this has the advantage of comprising a reasonably well-formed peer review process, the distillation process is still informal. Box and line notations, although widely employed, generally lack any cognitive underpinnings that might justify either their form or use. Since design knowledge usually needs to be accumulated and analysed over time, none of this is necessarily unreasonable. However, if viewed as a foundation layer for developing a more scientific basis for thinking about design, then it can only be regarded as being at best a rather ‘ad hoc’ layer. Furthermore, observational studies have tended to indicate that designers do not necessarily actually follow the design strategies in detail, even when they believe that this is what they are doing (Visser, 1987).

The argument for this position paper is therefore that any progress towards a ‘science of design’ must improve these foundations, and hence that ways of extending our empirical knowledge about design practices, and improving the quality of that knowledge are needed. In particular, we suggest that this may be best achieved by employing well-structured observational methods, in order to create a sounder foundation for understanding both the processes involved and the basis for ‘good’ design decisions, together with systematic approaches to synthesising the resulting data. Without such a systematic foundation, software design knowledge can only continue to be created and transferred by practices that are essentially craftsman-apprentice in their nature, rather than using engineering forms that incorporate any forms of ‘scientific’ evidence.

Limitations of existing design practices

As indicated above, knowledge about software design ideas is usually shared and transferred by practices such as design methods, design patterns, and architectural patterns. The forms of these can vary considerably (a ‘structured’ design method such as SSADM has very little in common with an agile method such as DSDM). Their forms are also constrained by the need to codify the procedures involved: the Unified Process used for OO design illustrates the problems of trying to describe a method that uses a non-linear set of procedures; while patterns are not easily indexed and classified. In addition, the forms of design activity required for many emerging

software technologies such as components and web services, involving reuse of existing software elements, are not adequately addressed by either methods or patterns.

Observational studies

Most of the observational work in this area has been conducted by psychologists, beginning with the seminal study by Adelson and Soloway (1985). These studies (reviewed more fully in (Budgen, 2003)) have demonstrated that, where designers are able to reuse their existing knowledge, the actual design activities are apt to be *opportunistic* (driven by the designer's evolving understanding of the problem rather than by any specific procedures), and involve the creation of quite rich 'mental models' which may then be 'executed' as the means of gaining further insight. Neither opportunistic processes nor mental execution of design models are readily incorporated into the design practices summarised in the preceding section. Although CASE tools offer some scope to assist here (and to help with managing complexity), these will need to evolve substantially to support such practices as component reuse.

The research agenda

There are two principal elements proposed in this position paper, which are as follows:

1. To foster the practices of empirical and observational studies of software design as the means of providing the basic 'knowledge layer' that software engineering needs in order to provide a firmer understanding of how to design software.
2. To find ways to codify, synthesise and promulgate the experiences distilled from the raw data, in such a way that others can readily access and employ these, and to develop tools that will assist with their use.

Incorporating these in a systematic framework would provide the basis for a more scientific approach to studying software design than has so far prevailed. (One should note here that we employ a fairly broad use of the term 'scientific' that includes forms such as cognitive science, ethnography etc.) Indeed, we would argue that if software engineering is to progress from a craft activity to engineering practices, this type of evolution is essential.

For the process of 'systemisation', we would advocate the development of a multidisciplinary forum that would equate to one of the 50 "Cochrane Groups" employed in the medical domain. Such groups act as a community that collectively reviews and systematically synthesises evidence relating to a specific topic, and then uses this to inform clinicians and to help develop and formulate clinical practice guidelines (Sackett, 1994; www.cochrane.org)

Author's background

David Budgen has been Professor of Software Engineering at Keele University since 1991, and is the author of many papers on software design as well as of the textbook *Software Design* (published 1993, second edition 2003). His research has addressed both the development and evaluation of software development environments and also the actual processes employed in designing software. He is director-designate of Keele's new *Centre for Evidence-Based Software Engineering*, a member of the EPSRC computing college since its foundation in 1997, and the BCS representative on the Steering Committee for the IEEE/ACM Software Engineering Curriculum.

References

- Adelson B. and Soloway E. (1985) The role of domain experience in software design, *IEEE Trans. Software Eng.*, **SE-11**(11), 1351-1360.
- Budgen D. (2003) *Software Design*, 2nd edition, Pearson Addison Wesley.
- Sackett D. L. (ed) (1994) *The Cochrane Collaboration Handbook*, Update Software, Oxford UK.
- Visser W. (1987) Strategies in programming programmable controllers: A field study on a professional programmer, in *Empirical Studies of Programmers: Second Workshop*, eds. Olsen G.M, Sheppard S. and Soloway E.), Ablex, 217-230.