Brady Field

Software Design: An Introduction

**Important points**

“The significant characteristic of design as a problem-solving approach is that there is rarely (indeed, almost never) only one solution to a problem. So we cannot hope to identify some systematic way of finding the answer, as occurs in the physical and mathematical sciences.” [209]

“Designing software is rarely a completely unconstrained process. The designer not only has to produce a solution to a given problem but must also meet other customer-imposed requirements. These constraints may include the need to design a solution that can be implemented in a particular programming language; or one that will work within a particular environment or operating system. Constraints therefore act to limit the ‘solution space’ that is available to the designer.” [210]

“A major need for the designer is to be able select and use a set of abstractions that describe those properties of the design model that are relevant to the design decisions that need to be made.” [211]

“There is a long tradition of drawing diagrams to provide abstractions in science and engineering, and even though the ‘invisibility’ factor makes the form of these less intuitive when used to describe software, they are still very useful.” [212]

“Statecharts provide a means of modeling the behavior of a system when viewed as a finite-state machine, while providing better scope for hierarchical decomposition and composition than is generally found in behavioural representation forms.” [213]

“The structural viewpoint is concerned with the physical properties of a design, and hence it is one that may need to describe many attributes of design elements. For this reason, no one single notation can effectively project all of the relevant relationships (such as encapsulation, scope of shared information, invocation), and so an effective description of the structural viewpoint is apt to involve the use of more than a single representation.” [214]

“Novice programmers can use the abstractions provided in a programming language to construct actual programs, and in the process receive feedback that can assist with revising their ideas and understanding during both compilation and execution of the programs.” [216]

“The choice of design strategy and associated method has significant implications for the resulting solution structure, or architecture.” [217]

“A characteristic of second-generation design methods is that they involve constructing much more complex design models from the start, usually involving the use of more than one design viewpoint.” [218]

“Overall, while our understanding of how software is designed is slowly improving, it seems likely that this will provide an active area of research for many years to come.” [219]

**Disagreements**

“The use of ‘methods’ for software design has no parallel in any other stage of software development. We do not have ‘testing methods’ or even ‘programming methods.’” [216]

There are methods we learn for both testing and development. Stub functions would be one example of a “programming method”. One could argue that test-driven development is an example of both a “testing method” and a “programming method”. I recognize that there is no “one size fits all” approach to either testing or development, but that is also true of design and every other aspect of software development. To say that these methods simply do not exist is incorrect.

**Questions**

I have no questions. I understood everything in the article.