Design Theory and Computer Science

Subrata Dasgupta

Cambridge Tracts in Theoretical Computer Science 15

DESIGN THEORY AND COMPUTER SCIENCE

Processes and Methodology of Computer Systems Design

SUBRATA DASGUPTA
Computer Science Trust Fund Eminent Scholar & Director
Institute of Cognitive Science
University of Louisiana at Lafayette

The right of the University of Cambridge to print and sell all manner of books was granted by Henry VIII in 1534. The University has printed and published continuously since 1584.

CAMBRIDGE UNIVERSITY PRESS
Cambridge
New York Port Chester Melbourne Sydney

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo, Delhi

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521118156

First published 1991
This digitally printed version 2009

A catalogue record for this publication is available from the British Library

ISBN 978-0-521-39021-7 hardback
ISBN 978-0-521-11815-6 paperback

DESIGN THEORY AND COMPUTER SCIENCE

Cambridge Tracts in Theoretical Computer Science

Titles in the series

1. G. Chaitin *Algorithmic Information Theory*
2. L.C. Paulson *Logic and Computation*
3. M. Spivey *Understanding Z*
4. G. Revesz *Lambda Calculus, Combinators and Logic Programming*
5. A. Ramsay *Formal Methods in Artificial Intelligence*
6. S. Vickers *Topology via Logic*
7. J-Y. Girard, Y. Lafont & P. Taylor *Proofs and Types*
8. J. Clifford *Formal Semantics & Pragmatics for Natural Language Processing*
9. M. Winslett *Updating Logical Databases*
10. K. McEvoy & J.V. Tucker (eds) *Theoretical Foundations of VLSI Design*
11. T.H. Tse *A Unifying Framework for Structured Systems and Development Models*
12. G. Brewka *Nonmonotonic Reasoning*
13. G. Smolka *Logic Programming over Polymorphically Order–Sorted Types*
15. S. Dasgupta *Design Theory and Computer Science*
17. J.C.M. Baeten (ed) *Applications of Process Algebra*
18. J.C.M. Baeten & W.P. Weijland *Process Algebra*

To my mother

Protima Dasgupta

When we mean to build,
We first survey the plot, then draw the model;
And when we see the figure of the house,
Then we must rate the cost of the erection;
Which if we find outweighs ability,
What do we then but draw anew the model
In fewer offices or at last desist
To build at all?

Henry IV, Part 2, I, iii

Though this be madness, yet there is method in 't

Hamlet, II, ii

Preface

In this book I intend to examine the logic and methodology of design from the perspective of computer science. Computers provide the context in two ways. Firstly, I shall be discussing the structure of design processes whereby computer systems are, or can be, designed. Secondly, there is the question of the role that computers can play in the design of artifacts in general – including other computer systems.

The aim of any systematic enquiry into a phenomenon is to uncover some intelligible structure or pattern underlying the phenomenon. It is precisely such patterns that we call *theories.* A theory that claims to explain must exhibit two vital properties. It must be *simpler* – in some well defined sense – than the phenomenon it purports to explain; and it must be *consistent* with whatever else we know or believe to be true about the universe in which the phenomenon is observed.

The phenomenon of interest in this book is such that it cannot be adequately described by a single sentence. That itself is an indicator of its inherent complexity – and therefore of its intrinsic interest. It is, perhaps, best described in terms of the following entities:

(a) *Computer systems.* I include in this term all nontrivial discrete computational devices (e.g., algorithms, logic circuits, computer architectures, operating systems, user-interfaces, formal languages and computer programs). Computer systems are characterized by the fact that they are artifacts; that they may be physical or abstract in form; and that, in general, they are complex entities.

(b) *Design processes.* These are characterized by the fact that they are cognitive and intellectual in nature. Design as an activity is, thus, psycho-biological in origin. It is a human activity.

(c) *Computer-aided design (CAD) systems.* These are also computer systems to which are assigned some of the tasks encountered during design. CAD systems are, thus, artifacts that either augment the cognitive/intellectual processes in design or, more ambitiously, attempt to mimic these same processes.

The central topic of this book – the phenomenon of interest – is the relationship among these entities. More specifically the question addressed here is the following:

Can we construct a theory of the design process – an explanatory model – that (a) can serve to clarify and enhance our understanding of how computer systems are, or can be, designed; and (b) consequently, provides a theoretical basis for building methods and computer-aided tools for the design of such systems?

Let us label this question 'Q'. There are at least two important issues that pertain to Q.

Firstly, it has been observed by many that the cognitive/intellectual activity we call design has a significant component that is domain-independent. Whether we are designing buildings, organizations, chemical plants or computers there are some principles or 'laws' that are common to all. Thus, quite independent of the specific design domain, it makes sense to talk of *general* theories of design – that is general, domain-independent explanatory models of the design process. The theoretical and intellectual value of any theory that we may propose in response to Q will, to a great extent, be determined by its generality – its domain-independence. A theory of design that is applicable to computer architecture, software and VLSI circuits is clearly preferable to one that is only applicable to VLSI circuits. A theory that is applicable to both computer systems and buildings is clearly more valuable than one that is only valid for buildings.

At the same time a theory of design is of heuristic value only when it provides advice on how to design specific systems within a specific domain. A grand theory of design is pointless if it is so general that no one knows how to relate it to specific problems. Thus, our search for a design theory must attend to both the theoretical need for generality and the practical quest for domain-specificity.

The second major issue relevant to Q is the debate on whether a theory of design is to be descriptive or prescriptive. A *descriptive* theory is an explanation of a given phenomenon as we observe it. All theories in the natural sciences are, of course, descriptive. When we enter the realm of artifacts – the realm of what Herbert Simon memorably termed the 'Sciences of the Artificial' – the issue becomes somewhat more problematic. For, given that design is a cognitive/intellectual process, it is clear that no design theory can afford to ignore or bypass the constraints imposed by human nature and intellect. To this extent, a theory of design must in part be descriptive. It must explain how design is conventionally carried out by humans.

In contrast, a *prescriptive* (or *normative*) theory is one that prescribes how something should be. Design is concerned with the making of artifacts – that is, entities that are in some well defined sense not natural; design is concerned with the purposive effecting of change. Thus, it is clear that a theory of design must have the capability of specifying how such change is best effected.

We can conclude that anyone embarking on constructing a theory of the design process must navigate cautiously between the Scylla of description and the Charybdis of prescription.

The urge to construct theories of design – to construct a logic of design – is neither new nor specific to the computer system domain. In particular, architectural design theory has a lineage that can at least be traced back to the Roman writer Vitruvius. One of the most celebrated treatises on the principles of architecture was by the 15th century Renaissance writer Leoni Alberti. In our own times, many architectural theorists and practitioners, including such pioneers such as Christopher Alexander and Christopher Jones, have pondered and written on the methodology and logic of their discipline, and I shall have occasion to refer to some of their ideas in this book.

In computer science[1] one of the earliest discussions of what we now call program correctness (an important aspect of computer systems design) is a relatively little known paper by Alan Turing published in 1949. Soon after, Maurice Wilkes's invention (circa 1951) of microprogramming must surely count as an important event in the methodology of computer design. In hardware logic design (or what is also called 'gate level' design) one of the debates of the late 1950s (as has been traced by Glen Langdon (1974) in his historical study of the discipline) was on a methodological issue. The so called 'Eastern School' favored the use of block diagrams in designing logic circuits while the 'Western School' advocated the use of boolean algebra. This is, in fact, a classic instance of the ever recurring debate between what might be called the 'naturalistic' and 'formalistic' schools of design methodology.

In computer science, design methodology really came of age in the mid 1960s when the problems of constructing and managing large scale software began to be openly and widely discussed. Perhaps the most influential figure from these times is Edsgar Dijkstra who in an important series of publications between 1965 and 1969 brought to our attention the intrinsic complexities attending programming and who prescribed

[1] In this book, I shall use the term 'computer science' to encompass all disciplines pertaining to computers and computing – including algorithms, languages, computer architecture, software, artificial intelligence, computer-aided design, VLSI design, etc. Similarly the term 'computer scientist' will refer to practitioners of any of these disciplines. I shall thereby avoid the tiresome distinction sometimes made between 'computer science' and 'computer engineering'.

techniques that were to crystallize into the, now well established, principles of structured programming. In this same period Robert Floyd and Tony Hoare published papers as a result of which the idea of programs as formally provable theorems in an axiomatized system was born. Contemporaneously, Herbert Simons's highly influential book *The Sciences of the Artificial* appeared in which the author presented the outline of what he termed a 'science of design'.

Since that very fruitful period the design process has become a subject of interest in all those areas of computer science where one has to come to terms with the problems of large scale complexity. These areas range from such relatively 'soft' areas as computer architecture and the design of human–computer interfaces to the relatively 'hard' domains such as microprogramming and VLSI circuit design. Finally, interest in design theory amongst computer scientists and amongst engineers and architects has been further sharpened by two computer related advances: computer-aided design and applied artificial intelligence.

If one examines the literature on design theory – both inside computer science and outside it – one encounters a small number of recurrent and closely intertwined themes. Is design art or science? Can we construct a genuine logic of design? Should we try to formalize the design process? What is the relationship between design and mathematics? What is the connection between design and science? Are designs computable? What is the nature of design knowledge?

These themes form, so to speak, the very stuff of this book. By addressing these and other questions I hope to draw the reader's attention to the enormously complicated phenomena surrounding the design act and their implications for design methodology and design automation. At the same time by attempting to respond to these issues within the framework of a systematic and coherent set of ideas I hope to shed some further light on the structure of design processes. This, as previously noted, is the primary aim of any theory of design.

This book consists of three parts. In Part I the fundamental characteristics of the design process are identified, discussed and analyzed. In the context of the description/prescription duality, Part I is descriptive in spirit and intent. I shall examine design as an activity that is 'out there' in the 'real' world – an activity that can be empirically studied and analyzed just as one studies any other empirical phenomenon. In the course of this discussion examples and illustrations will be drawn from various types of computer systems, notably computer architectures, operating systems, logic and VLSI circuits and user-interfaces. However, since many of the ideas discussed in Part I also apply to other 'Sciences of the Artificial' – specially engineering and architecture – I shall also have occasion to refer to the work of design theorists in these other disciplines.

Part II is wholly prescriptive in spirit and intent. It is concerned with *design paradigms* – that is, specific philosophies of, or approaches to, design (with or without the assistance of computers). Obviously, any prescription that one may make about what the design process *should be* – that is, any design paradigm that one may invent – is severely constrained by *what is possible*. Thus, the characteristics discussed in Part I establish the framework within which the various paradigms appearing in Part II are analyzed, criticized or recommended.

To this writer, the most interesting question in design theory is the relationship between design and those two great intellectual enterprises, mathematics and science. A substantial portion of Part II is devoted to design paradigms that are explicitly or implicitly modeled on the relationship between design, mathematics and science. However, if there is a single thesis that this book may claim to advocate, it is that from the perspective of methodology, one may actually conduct design in a manner that makes it indistinguishable from the activity we call science. Part III is concerned entirely with arguments and evidence in support of this thesis and its consequences, especially in the realm of computer-aided design.

Acknowledgements

Quite apart from the hundreds of authors cited in the text, I owe a massive debt of gratitude to many individuals and organizations who, in one way or another, have influenced the final shape of this work. In particular, I thank the following:

- Tony Hoare (Oxford University), Werner Damm (University of Oldenberg, Germany) and B. Chandrasekaran (Ohio State University) for their various and very individualistic insights into the design process.
- Bimal Matilal (Oxford University) and James Fetzer (University of Minnesota) – two philosophers – for discussions or correspondences regarding matters philosophical.
- Karl Klingsheim (University of Trondheim and Elektroniklaboratoriet ved NTH, Norway) – engineer turned design philosopher – who suffered my theorizing with patience and good humour.
- Sukesh Patel, Ulises Aguero, Alan Hooton and Philip Wilsey – wonderful collaborators and former students.
- N.A. Ramakrishna – my research assistant without whose heroic help the manuscript would never have been finished.

The book was conceived during a summer spent at the University of Oldenberg, Germany in 1988. I am deeply grateful to the German Science Foundation for supporting me with a Guest Professorship and to Werner Damm and members of his 'group' for their intellectual, social and logistical support.

I thank the US National Science Foundation for their award of a grant for the period 1988–9 and the University of Southwestern Louisiana (USL) for a Summer faculty award in 1989. As ever, I am grateful to USL and, in particular, the Center for Advanced Computer Studies for the unusual, flexible and collegial atmosphere that is so conducive to projects such as this.

I thank Keith van Rijsbergen (Glasgow University) – the editor of the series in which this book is being published – and David Tranah (Cambridge University Press) for their initial, warm support of this project. And, of course, my editor at Cambridge University Press, Alan Harvey, for his continuing, friendly and patient help throughout the production process.

I thank the following for granting me permission to use material from their publications:

- Harvard University Press, for the quote from C. Alexander, *Notes on the Synthesis of Form* (1964), p. 1, appearing on page 3.
- John Wiley and Sons, for the excerpts from J.C. Jones, *Design Methods: Seeds of Human Future* (1980), p. 64 appearing on page 145, and from Y.E. Kalay (Ed), *Computability of Design* (1987), p. xi, appearing on page 2.
- Springer-Verlag, for adaptation of material from S. Alagic and M.A. Arbib, *The Design of Well-Structured and Correct Programs*, pp. 124–7, appearing on pages 83–91.
- The Institute of Electrical and Electronic Engineers, for material from the papers: (a) 'Domain Specific Automatic Programming' by D.R. Barstow, *IEEE Transactions on Software Engineering,* Vol. SE-11, #11, Nov. 1985, pp. 1321–36. (b) 'Understanding and Automating Algorithm Design' by E. Kant, *IEEE Transactions on Software Engineering,* Vol. SE-11, #11, Nov. 1985, pp. 1361–74. (c) 'Automatic Data Path Synthesis', by D.E. Thomas *et al., Computer*, Vol. 16, #12, Dec. 1983, pp. 59–70.
- Lissa Pollacia and Lisa Levy for excerpts from their project report 'PDI Constraints' (USL Center for Advanced Computer Studies, April 1988).

Finally, my thanks to my wife Sarmistha and sons Jaideep and Monish. As always, they suffered the domestic consequences of this enterprise with phlegmatic forbearance.

Contents