CHAPTER ONE

THE ROLE OF FORMAL METHODS IN HUMAN-COMPUTER INTERACTION

MICHAEL HARRISON AND HAROLD THIMBLEBY

1.1 Introduction

In many respects Human-Computer Interaction, HCI, is a discipline that is still in its infancy; its growth to maturity is stunted because it is the child of the marriage of two rather disparate subjects, psychology and computing. This book contains a collection of contributions that have in common the belief that the subject's multidisciplinary nature, which at the moment polarises and hampers its development, will eventually lead to a rich and significant body of knowledge. Uncertainty about HCI's parentage, whether psychological or computational, tends to lead to work which is adopted by one heritage to such an extent that it is not easily recognised by the other. Such rather biased genealogies in practice do little to aid understanding of *what* makes an effective human-computer system and *how* to systematically design, implement, evaluate, improve and maintain one.

We believe that much of the early work in HCI has been encumbered by a lack of appropriate abstractness or applicability to the design process; so research results are too specific to be generally applicable,

or perhaps too abstract to be applicable at all, or relate to aspects of user behaviour without providing information that can be used in design (before the user can use the system). There is a dilemma that design comes 'first' and use 'last,' and the effectivness of a working system depends on many criteria that cannot be anticipated at the early stages of design—for instance, how introduction of a computer system changes a user's error rate, productivity or job satisfaction. On the one hand, such details quickly overwhelm designers and are likely to be ignored; on the other hand, though abstraction promises a way out of this complexity, so far HCI as a discipline has not progressed sufficiently for a designer to know what to abstract and what to represent—nor how to reason about the design in abstract form.

Why *formal methods* in HCI? HCI is a multidisciplinary activity. It is difficult to express the contributions of psychology, sociology, software engineering and so on, in forms that are understood properly and contextually by all parties to the design process. In order to understand the contributions of the different approaches and their proper bearing on design we need to be precise about scope and role. The challenge of formal methods (precise notations and mathematical models) in interactive system design and human computer interaction is to produce a precise framework in which the role and scope of these models may be clearly understood. It is possible for people to make accidental or deliberately outrageous claims that in fact have little substance; indeed, this happens rather too often. It is difficult to assess such claims. Any system is 'easy to use' under the right circumstances (say, when it is used by its designer!)—and other workers interested in achieving the same effects probably do not have the resources to check out the claim *unless* it can be expressed formally.

1.2 Some examples

Formal methods first specify *what* we are talking about and lay down precise rules about *how* one is allowed to reason about those things. Three examples will illustrate these ideas (the last two are also to be found elsewhere in this book). They are intended to demonstrate how formal methods are to be developed to describe different aspects of the HCI problem and demonstrate strengths and weaknesses of current approaches.

Time

Perhaps an extreme example will initially illustrate the idea: suppose we decide to formalise the notion of **response time**. How fast does and how fast should, the computer respond to the user's actions?

A decision to formalise issues of time naturally restricts us; we cannot make claims about noise, or colour, or command syntax being relevant. On the other hand, we gain a whole body of mathematics, for instance we know how to add, average, and perform all sorts of reliable, standard statistical operations on response times. So long as we are happy being restricted to such an abstract level of discussion, and very carefully circumscribe our claims, then our results are quite certain. Furthermore, the formal process can be written down, and communicated precisely (possibly after training) to other workers, designers or evaluators. Anything written as a formal statement can be, and is *intended* to be, criticised.

As a research issue: if we find some factor of response time that is essential for usability (in terms of response times) but which we cannot formalise as a measure of time, then we have discovered something very important that will impinge on the whole programme of formalising HCI and the design of interactive systems. This situation is analogous to the hypothetical discovery in physics that, say, the gravitational attraction of two bodies depends not just on their mass but also on their flavour. It is curious that we can obtain *general* abstract notions such as force in the natural world, but we have, so far, sought vainly for them in HCI.

Competence

As a contrast with the precise framework described in the previous subsection, a formalisation of the **competence** of the user is certainly more difficult to be precise about. We can formalise the notion of task, the actions that the user must carry out in order to complete the task, and perhaps observe that the nature of consistency in understanding the functionality of a system corresponds to regularities in action sequences. So-called Task Action Grammar (TAG, described in Chapter 2) thereby provides a structure for a notion of competence. However it is not clear how such a structure and analysis could be used prescriptively to produce acceptable consistency. Despite these reservations, TAG provides a clarity in the design process that aids an understanding of

the concepts involved, and may provide a means for preventing the familiar *ad hoc* accretion of features in interactive systems.

Determinism

A third example of models that bridge the gulf between user and system is discussed in Chapter 4, where it is shown that **determinism** is a phenomenon that occurs at several levels in interaction. A deterministic system may *appear* non-deterministic to a user because the user has incomplete information about how the system works. Formal descriptions of the system to reason about real and apparent non-determinism in the design of the system are relatively easy to formulate, and highlight and inter-relate many interesting issues.

1.3 The scope of formal methods

Formal methods, then, provide a clarity which is much needed in HCI. With some mathematical training, one can enter into *rigorous* debates about design issues. It is not necessary to have a certain sort of system, nor is it necessary to take experimental steps to control for interpersonal variation (differences between users) or subjective opinions.

The main difficulty of formal methods in HCI, which this book starts to address, is the limitation in their scope. If discussion about interactive systems is restricted sufficiently to be able to talk about them formally, then many important features that *may* affect the user are lost. For example, response time can be talked about formally, but this discussion does not include the effect, for instance, of the hourly-rate wages on the user's strategy for delaying the system. This is a fundamental tradeoff: the balance between being precise and being able to be precise about aspects of interaction that are truly significant.

But the effort to develop formal methods is worth making, as may be highlighted by viewing, say, typical high volume commercial software products such as spreadsheets. It is clear that visible features sell systems and that, currently, usability features (or, rather, usability *claims*) are in vogue. This notion of usability is often communicated to a potential buyer by stressing design features and styles such as WYSIWYG, direct manipulation, the use of windows etc. These blanket notions are currently only loosely understood—or, at least, we do understand some aspects of the ideas, but are not yet agreed what the essence of

the notions really are or when these notions will produce improvement in usability. Formal methods should cut to the essence.

We should start asking, not what can we formalise, but what is worth formalising? And thereby expose and clarify what the essence of such interactive styles really is. Generally, more precision is required in the description of styles, direct manipulation and so on, and many other features of interactive systems. We need to explain to designers and implementers what they must or could do in order to make their interactive system adequately directly manipulable, or whatever. We also need to understand what makes a certain style (e.g., direct manipulation) applied in a particular system and context, more truly usable. We believe that a more precise understanding of these issues will not only improve future generations of design but will also give precise shape to features of design that may be used to control evaluation of the usability of interactive systems incorporating these styles.

There are many technical approaches to formal methods; the choice of approach is tailored to the design context and the sort of design decisions that are required. Perhaps one of the most exciting aspects of formal methods is that of *insight*: suddenly, precise reasoning is not so much onerous as creative. A formal model may suggest new ways of looking at a system design, new ways of evaluating it, new ways of inventing features and new ways of improving human computer interaction.

1.4 Book structure

This book aims to contribute to HCI by exposing a range of formal methods. We discuss ways of improving our understanding of both the concepts of interaction and the process of design of interactive systems. Hence, formal methods provide a framework within which the concepts of user, system and their interaction may be made precise. Our purpose in providing this framework is to underpin the design and implementation process. A conceptual framework without predictive or evaluative functions and which cannot eventually be transferred to the everyday domains of application design is of no value. Hence this framework should be sufficiently tractable to make it possible to reason about design; and sufficiently pragmatic to guide the design process. Several of the chapters illustrate this point in more detail:

INSIGHT: *the gulf between software engineering and human computer interaction.*

The two initial chapters in this section take two standpoints. Schiele and Green's chapter takes a "psychological engineering" perspective of *consistency* in interactive systems. They are concerned with psychological structures; whether a psychologically valid notion of consistency may be used to analyse or predict the structure of user interfaces for commercial scale interactive systems. Took on the other hand is rooted in the traditions of software engineering. He is concerned principally with functionality and computational issues such as *completeness* and *consistency.* The fact that he is developing an interactive system leads him to propose constraints that are not only *hard* (i.e., based on implementation) but also *soft* (i.e., dependent on more normative concerns). He develops extensions to the notions of completeness and consistency within the Z specification method that encompass intuitive ideas of usefulness or usability. Such notions include *power* or *equippedness* of the interface. Z is particularly interesting because it is a method that has been finding increased industrial acceptance in the UK over the last few years.

Bridging the gulf then involves recognising the software engineering implications of a psychologically derived notion of consistency based in a notation such as Task Action Grammar, and the psychological implications of the soft constraints based on consistency and completeness contained in Took's Chapter 3. These two chapters establish the agenda for the rest of the book.

MODELLING: *interactive behaviour and applications to the design activity.*

The middle chapters of the book develop various models of interactive behaviour that allow designers to reason about the interactive behaviour of the system from the user's point of view. Many representations exist that describe the *user computer system* from different perspectives, but here the authors are concerned with abstract models of interactive behaviour of the system and their relation to the design process. These models may be used by the designer as early descriptions of the emerging design. The three chapters have strong similarities. One of them (Sufrin and He, Chapter 6) describes the trace behaviour of an interactive system using Z. This model is interesting both because it demonstrates the possibility of formulating user-engineering principles

within an accepted engineering framework, and because it demonstrates that notations like Z may be used as a basis for reasoning.

The other two chapters use mathematical frameworks to provide models of important issues in the design of *user computer systems.* Dix, Chapter 4, tackles the problem of non-determinism. It notes that different notions are relevant to actual non-determinism in the system and perceived non-determinism in the user. Harrison & Dix, Chapter 5, develops a state-display model of interactive behaviour that is appropriate for the modelling of direct manipulation characteristics of interactive systems.

REFINEMENT: *models to prototypes.*

The later chapters either describe the transition from abstract models to rapidly developed prototypes, or from abstract models to production systems. A model of interactive behaviour provides a designer of interactive systems with an uncluttered description to be used as a framework for discussing design alternatives; furthermore, a mathematical model may also be used as a means of constraining implementation to be consistent with respect to stated design concepts or principles. Such analysis and discussion is useful in helping the designer to reflect on design from a more user-orientated perspective. However, ultimately the final arbiter of a successful design is the user in the context of actual use of the system.

Methods are therefore required to produce executable representations of models that clearly reflect the principles used in design. This part of the book contains three chapters. Runciman and Alexander, Chapters 7 and 9, address directly the problems of rapid prototyping of interactive systems; the Cockton, Chapter 8, considers issues related to the structure of an architecture for interactive systems based on formal models. These chapters point to a number of problems that must be solved before an effective methodology for the rapid development of interactive systems may be derived from models.

1.5 The future

It is tempting to assess the future possibilities of a synthesis of Formal Methods and HCI. With increasing appreciation of the benefits of formal methods in HCI, the two subjects will develop symbiotically. Formal methods will become easier to use and therefore more widely

relevant outside the academic community; and HCI will become better, more coherent and easier to apply in the critical early stages of design. We hope that this collection of research contributions will positively support this symbiosis.

CHAPTER TWO

HCI FORMALISMS AND COGNITIVE PSYCHOLOGY: THE CASE OF TASK-ACTION GRAMMAR

FRANZ SCHIELE AND THOMAS GREEN

2.1 Introduction

Formal methods already have a recognised place in software engineering. The arguments for using formal descriptions and specifications in that context are familiar and are frequently rehearsed again for the benefit of apprentices: they bring precision of meaning, and they allow quasi-algebraic techniques of manipulation, verification, etc. Bringing them into the realm of Human-Computer Interaction (HCI) can contribute similar benefits.

However, bringing formal methods into HCI can do more than just extend their domain. When an interface has been formally described, we can potentially apply metrics to that description and *predict* whether the interface will be easy to use. An interface whose description is more complex, according to some criterion, will be harder to use. This formalist approach might well turn out to be helpful in design and evaluation. Particularly appealing is the fact that formal descriptions

might *reveal the inner workings* of the systems they describe rather than the surface characteristics, since it is the inner workings that are hardest to evaluate by traditional human factors approaches. But to achieve this ambition, the formal description and the metrics of complexity must be based on psychologically valid principles.

The authors now regard that platform with some scepticism. In the first place, we believe that very few methods for the formal description of interfaces even attempt to describe things from the user's point of view. Second, even where such attempts have been made, we believe they do not possess adequate structural power to describe the user's conceptualisations. This point has been developed in a review of formalisable models of user knowledge [77] where we discuss a number of such methods, including GOMS [33] and 'Cognitive Complexity Theory' [113], and illustrate some of their shortcomings. And third, we now believe that formal methods must either accept sharply-defined boundaries of applicability, or else be replaced by conceptual modelling techniques.

We shall consider in detail the case of Task-Action Grammar (TAG) [149], a formalism which presents a psychologically-grounded analysis of the property of *consistency*. In Green et al. [77] we show that TAG has greater structural power, and is better able than earlier formalisms to describe those properties of the user's conceptualisation which make an interface appear consistent or inconsistent. Because it has greater expressive power, and because this power is based on known phenomena of cognitive psychology, it has succeeded in making correct predictions of performance in some contexts; it has also been the starting-point for some spin-off developments. On the other hand, when applied to real applications programs, we have found ourselves juggling with the constraints of the formalism and being forced to invent ingenious solutions to local difficulties, rather than, as we would ideally have found ourselves, restfully choosing from alternative representations the one that most clearly fitted our intuitions.

This case study illustrates several propositions. On the plus side, formal methods can yield successful performance predictions and reveal flaws in the design of a user interface, as long as they are psychologically-grounded and possess adequate descriptive power. They can also lead to interesting spin-off developments such as described below (see §2.4). On the minus side, formal methods can lead to preoccupation with de-