Chapter 2
Topics of Formal Methods in HCI

Judy Bowen, Alan Dix, Philippe Palanque and Benjamin Weyers

Abstract In this chapter, we present an overview of some of the general themes and topics that can be seen in research into formal methods in human–computer interaction. We discuss how the contents of the rest of the book relate to these topics. In particular, we show how themes have evolved into particular branches of research and where the book contents fit with this. We also discuss the areas of research that are relevant, but are not represented within the book chapters.

2.1 Introduction

This chapters of this book are organised into three sections: modelling, execution, and simulation; analysis, validation, and verification; and future opportunities and developments. These represent specific themes of intended use under which we can group the work presented. While these (somewhat) broad groupings provide a particular categorisation, there are wider topics of interest we can describe when we

J. Bowen (✉)
University of Waikato, Hamilton, New Zealand
e-mail: jbowen@waikato.ac.nz

A. Dix
School of Computer Science, University of Birmingham, Birmingham, UK
e-mail: alanjohndix@gmail.com

A. Dix
Talis Ltd. Birmingham, Birmingham, UK

P. Palanque
IRIT—Interactive Critical Systems Group, University of Toulouse 3—Paul Sabatier, Toulouse, France
e-mail: palanque@irit.fr

B. Weyers
Visual Computing Institute—Virtual Reality & Immersive Visualization, RWTH Aachen University, Aachen, Germany
e-mail: weyers@vr.rwth-aachen.de

B. Weyers et al. (eds.), *The Handbook of Formal Methods in Human-Computer Interaction*, Human-Computer Interaction Series,
DOI 10.1007/978-3-319-51838-1_2

consider the literature of formal methods in HCI, under which the chapters of this book can also be considered.

HCI is in itself a broad topic and covers aspects of humans and their cognitive and physical capabilities, as well as machines and interface design at varying levels of detail. HCI must also consider the combination of these two things. When we introduce formal methods into the process, they may similarly be used for some, or all, of these considerations. Formal methods are also used in different ways across the HCI topics, and this is typically driven by the rationale behind the method being used and the purpose of its use within the process. While the benefits of using formal methods across all parts of the design process may be more obvious when working in safety-critical domains (health care, finance, transport, power generation, to name but a few), there is much to be gained from specific applications of use in other domains too, particularly when we consider the increasing levels of sophistication of both interfaces and interaction techniques as well as the ubiquity of such systems in the modern world. We can see examples of each of these approaches in the chapters that follow.

While it may appear that the differences between the HCI practitioner and the formal method practitioner are many, this is primarily an artefact of the approaches they use. In fact, both approaches have a common goal: the ability to reason about the software under construction in order to ensure it satisfies some set of requirements. Whether these requirements are expressed formally (as a specification for example) or as a set of user goals and tasks is then irrelevant, the point is to have some mechanism for ensuring these are met, and it is here that the use of formal methods for HCI becomes more obvious.

In the early years of research into formal methods for HCI, several approaches were developed which were based upon the use of existing formal methods and languages which were then applied to the topics of UI design and HCI, for example Jacky's use of Z to describe the interface to a radiation machine (Jacky 1997). Specialised approaches for interactive elements were developed, but still based on existing formalisms, such as the description of interactors in Z (Bramwell et al. 1995), VDM (Doherty and Harrison 1997), Lotos (Paternò et al. 1995), etc. This subsequently led to the development of new formalisms, or extensions to existing languages and notations, which were more suited for considerations of the user interface, interactions, and human users. While the design process for interfaces and interactions typically focuses on creating shared artefacts that can be used to communicate with all stakeholders in the process (as we would see in a typical user-centred design approach), this does not mesh naturally with the notations required for a more formal process.

Research into formal methods for HCI provides ways of supporting the development of interactive systems at differing levels of rigour and formality so that we gain the advantages that come from a more formal approach, while recognising the differences in process required when we must also consider users and interactions. While the nature of interfaces and types of interaction has changed considerably in the intervening years, many of the crucial factors in ensuring reliability, usability, safety, and soundness (e.g.) have not.

In this chapter, we consider topics of formal methods and HCI more generally and discuss how this has influenced work in the domain. We also describe how the chapters presented in the rest of the book contribute to these themes and build on the existing body of work.

2.2 Describing the Human User of Interactive Systems

HCI methods used to understand the human user have many of their roots in the disciplines of psychology, ergonomics, and pedagogy, as they seek to understand human capabilities (cognitive and physical) in order to design interactive systems that are usable and learnable, and which support the required tasks. While the complexity of human thought and behaviour in combination with proposed interactions may seem like a good fit for the use of formal methods, it is not without its challenges. One of the main problems faced when trying to formalise the user and their behaviour is that of unpredictability: we can never be certain what the user will do, and we can only surmise how they might behave. Typically then, most approaches abstract the problem by defining particular aspects of human behaviour rather than trying to capture full details of a user's thought process, understanding, motivations, and memory. These more abstract representations of users can then be used in conjunction with a model of a UI, or proposed set of interactions, to try and find areas of system interaction that may be problematic for a user and therefore more likely to lead to erroneous behaviour.

In Curzon and Blandford (2002), Curzon and Blandford used formal models of user cognition and actions. These are defined as rational actions a user will take to achieve a goal (so abstract away from random or malicious behaviours). The method is concerned with generic user models and examines how users may make mistakes in particular interfaces and how specific design rules would prevent this. This work has been developed over the years (and has its own basis in the earlier programmable user model concepts (Butterworth and Blandford 1997)), forming the foundation for several research approaches to user modelling. An example can be seen in Chap. 8, which builds on these models to develop the notion of user salience and then combines this with activation theory to try and predict likelihood of user error for particular designs. This enables a comparison of different designs so that those less likely to be problematic for users can be selected.

While the approach of Chap. 8 considers the user in terms of actions that are related to their goals and tasks, these are distinct from any model of the interactive system (allowing different combinations to then be considered). In contrast, Chap. 13 shows how human behaviour can be incorporated into larger formal models of the system to support verification. Here, the user behaviour (or the behaviour of several users) is described as a collection of tasks which are composed of a hierarchy of activities and actions. This is an example of how a well-used HCI technique (task analysis) can be enhanced through a formal approach (we will see the use of task

analysis in several of the chapters) and then subsequently be used to build larger combined models of the system and interface/interactions (a theme that is revisited in Chap. 6).

Another use of task analysis, in this case task decomposition, is presented in Chap. 11 which describes how to create mental models from task decompositions for the purpose of assisting users in learning and using an interactive system. Again, there is an abstraction of user behaviour, in this case into rational task decomposition. However, this is different from the approach mentioned above in terms of the use of formal methods. Rather than using them to support design decisions or system verification, here the unambiguity supports a structured approach to guiding the use of the system. In common with the work above though, the starting point is the human user and the actions they can perform.

2.3 Formal Methods for Specific Types of Interactive Systems

Although we talk about interactive systems collectively as if they are all essentially the same, there are of course vast differences between these systems, ranging from single-user desktop-based systems to multi-user mobile adaptive systems, and everything in between. These differences, and their inherent complexities, have led to another strand of research in the use of formal methods for interactive systems which is to assist with reasoning about these different types of interface, interaction, and use type. We still see elements of user modelling in this work (particularly where the concern is multi-user systems), but typically the primary focus is on aspects of the interaction design itself.

Adaptive, context-aware, and plastic interfaces are good examples of categories of interface that lend themselves to formal analysis. The designer must be able to reason about how, and when, the interface changes to suit its context of use or platform. These types of UI are the focus of several different approaches. We see some examples of this in Chaps. 5, 7, 12, and 18 which demonstrate the range of different methods which can be used in this area. The notion of transformation of interfaces (either for plasticity or adaptivity) has a long history, much of it related to the use of XML-based languages such as XIML (Puerta and Eisenstein 2002) or USIXML (Limbourg et al. 2004) which have subsequently been incorporated into larger groups of tools (e.g. Michotte and Vanderdonckt 2008) or development frameworks such as TERESA (Correani et al. 2004).

There are other transformation approaches that have been developed, some based on more traditional refinement concepts from formal methods (see, e.g. Bowen and Reeves 2009; Oliveira et al. 2015) and others which combine several approaches, such as that of Chap. 10 which is based on Petri nets, category theory, and graph rewrite rules. There are also considerations beyond the interface which must be reasoned about when we investigate context-aware systems, and works such as Abi-Aad

et al. (2003) and Costa et al. (2006) address this by describing models of the context itself, which can then be used in combination with system models. A different approach to formally modelling the context of use is also demonstrated in Chap. 12 where it is the combination of context model with interaction model that is used to determine suitability of a system under a particular set of circumstances.

While consideration of specific types of interactive systems is based on properties of the system itself, we can also consider types of UI under the umbrella of their use-domain, for example safety-critical systems. While these may be intended for use in very different use-scenarios (aircraft cockpits, medical devices, driverless trains, banking systems, etc.), they share common requirements. There are aspects of these systems which can potentially lead to serious loss or harm and as such we apply formal methods to such systems in order to reason about the safety criteria.

Although the focus of the chapters in this book is on the case studies presented in Chap. 4 (two of which are, of course, safety-critical applications), several of the chapters describe work which has been used in other safety-critical domains. Chapters 6, 8, 12, 13, and 14, for example, present work which has been used with medical devices, and the methods described in Chaps. 17 and 20 have been used in aircraft cockpit specifications.

Not all systems and interfaces are used by individuals. In complex systems, it is common for several users to work on individual interfaces which are all part of the same system. Systems which enable collaboration of users (or which provided mechanisms for collaboration) are sometimes termed ‘groupware’ or more commonly now ‘multi-user systems’. These come under the umbrella of ‘computer-supported cooperative work’ (CSCW) where the users, and use, are often considered along dimensions such as synchronous or asynchronous and colocated or remote. This matrix, first described in detail in Baecker et al. (1995), is the basis for several frameworks which were developed to help manage the design of such systems, see Greenberg (1996), Guicking et al. (2005), Antonaya and Santos (2010) for example.

Dealing with several users and multiple interfaces increases the difficulty when trying to ensure the system will behave as expected. As such, it is not surprising that this attracts the attention of formal practitioners. An example of such a system is seen in Chap. 15 where multiple interfaces in an air traffic control domain are modelled as multi-agent systems to consider human–machine, human–human, and machine–machine interactions.

Another way to approach this problem is to develop interface models which are modular and can therefore be composed (to develop the sort of system above) or exchanged and reused to either develop new systems from existing components or update parts of a system and be sure that certain behaviour is preserved. An example is shown in Chap. 5 where the appearance elements of the interface and the interaction logic are modelled separately to enable either (or both) to be considered as components of a larger system.

A factor of these more complex systems is that there are often elements of automation at play, which can themselves be considered as a type of interaction, and must certainly be reasoned about, in conjunction with the human user, in order to fully understand how such a system might behave. Chapter 7 discusses this showing how

explicit system automated behaviours can be identified using activity modelling, whereas in Chap. 19, we see how interconnection of small interactive systems (in this case apps) can be described to consider how they might usefully be connected.

2.4 Descriptions of the Modelling Process and Supporting Tools

Not all formal modelling is aimed at specific design purposes as shown above. There is a body of work where the focus is on suitable models for interactive systems (the hows, whys, and wherefores) where the models can be used more generally across the design process. Some of these focus on aspects such as model-checking, verification, and validation with the aim of showing how these can be applied in the domain of interactive systems. These can be used for ensuring safety properties are met (in critical systems, for example Loer and Harrison 2002; Bowen and Reeves 2013) or to explicitly manage error-prone interaction types that may occur in many systems, such as number entry (Thimbleby 2015).

These general modelling approaches have to tackle the problem of separation of concerns between interface and functional elements, while at the same time managing the relationship between the two. This can be done either by explicitly separating the two parts within the modelling (as we see, for example, in Chaps. 6 and 14), or by focussing on properties of the interactive components which can then be considered as a separate entity, as in the approach shown in Chap. 9.

Going beyond the standard formal approaches of model-checking and verification and moving into the domain of code generation has also been considered. In some ways, UIs are well suited for such automation as specific widgets and layout types can be derived from models (see, for example Eisenstein and Puerta 2000). However, there are also known problems with fully automating the interface design process as typically the aesthetics suffer and usability and appearance may also be compromised.

It can be problematic to introduce new languages into the design process as they may not suitable for supporting the necessary collaboration between design team and end-user. As such, visual notations or domain-specific languages may be more suited in these environments. Chapter 16 addresses this problem by discussing domain-specific languages and shows how these can be used in interactive system development as a more intuitive solution for experts in the field (as opposed to experts in formal methods). Similarly, the use of support tools to simplify the inclusion of formal methods into design is another way to try and reduce the gulf of understanding. This is seen in the earlier discussed work on XML-based languages which incorporates a number of different design tools to make such integration easier and is represented here in Chap. 18 which describes tools which support task and interface modelling at different levels of abstraction.

2.5 Summary

Finally, beyond the practical modelling approaches, tools, and methods which are presented, there is also reflective work which seeks to consider some of the larger encompassing challenges that all such work must address. For example, just as the interactive systems we develop must be useful and usable for their intended use-groups, so too the methods and design models we create must similarly be useful within the design process and understandable by designers. The models and formalisms we use also have different types of users (designers, developers, end-users, etc.). We must also ensure that they are appropriate for each user and perhaps provide different views (via visualisations for end-users of specifications for software engineers e.g.) of the models created. Two chapters in this book address this issue, Chap. 17 from the perspective of usability of verification tools, while Chap. 20 considers the gaps which may still be present, however thorough the specification and verification process may be.

The characterisations of the work presented in this book via the topics of the four sections are not necessarily that clear cut. Similarly, the work discussed in this chapter encompasses many different topics of the book chapters rather than being neatly segregated as described. The intention is to show some of the common themes that exist in formal methods and HCI research and provide an overview of how the chapters support these, rather than imply that all of the work fits exactly under just these headings.

Not all of the topics and areas that can be identified are represented in this book. This is hardly surprising as the domains of both HCI and formal methods continue to grow and expand almost as quickly as the systems they describe. Some of the areas not covered here can be found in the proceedings of relevant conferences such as (EICS, INTERACT, CHI, FM, ICFEM, etc.) and are also discussed further in Chap. 3.

References

Abi-Aad R, Sinnig D, Radhakrishnan T, Seffah A (2003) CoU: context of use model for user interface designing. In: Proceedings of HCI international 2003, vol 4, LEA, pp 8–12

Antonaya SL, Santos C (2010) Towards a framework for the development of CSCW systems. In: Luo Y (ed) Cooperative design, visualization, and engineering, vol 6240. Lecture Notes in Computer Science. Springer, Berlin Heidelberg, pp 117–120

Baecker RM, Grudin J, Buxton WAS, Greenberg S (eds) (1995) Readings in human-computer interaction: toward the year 2000, 2nd edn. Morgan Kaufmann

Bowen J, Reeves S (2009) Supporting multi-path UI development with vertical refinement. In: 20th Australian software engineering conference (ASWEC 2009), 14–17 April 2009. Gold Cost, Australia, pp 64–72

Bowen J, Reeves S (2013) Modelling safety properties of interactive medical systems. In: Proceedings of the 5th ACM SIGCHI symposium on engineering interactive computing systems, ACM, New York, NY, USA, EICS '13, pp 91–100. doi:10.1145/2480296.2480314

Bramwell C, Fields RE, Harrison MD (1995) Exploring design options rationally. In: Palanque P, Bastide R (eds) Design, specification and verification of interactive systems '95. Springer, Wien, pp 134–148

Butterworth R, Blandford A (1997) Programmable user models: the story so far. Puma working paper WP8, Middlesex University

Correani F, Mori G, Paternò FM (2004) Supporting flexible development of multi-device interfaces. In: EHCI/DS-VIS, pp 346–362

Costa PD, Guizzardi G, Almeida JPA, Pires LF, van Sinderen M (2006) Situations in conceptual modeling of context. In: EDOC workshops, pp 6

Curzon P, Blandford A (2002) From a formal user model to design rules. In: Goos G, Hartmanis J, van Leeuwen J (eds) Interactive systems: design, specification and verification, no. 2545 in Lecture Notes in Computer Science, Springer Berlin, pp 1–15

Doherty GJ, Harrison MD (1997) A representational approach to the specification of presentations. Eurographics workshop on design specification and verification of interactive systems, DSVIS 97. Granada, Spain, pp 273–290

Eisenstein J, Puerta A (2000) Adaptation in automated user-interface design. In: IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces, ACM Press, pp 74–81

Guicking A, Tandler P, Avgeriou P (2005) Agilo: a highly flexible groupware framework. In: Groupware: design, implementation, and use. In: Proceedings of 11th international workshop, CRIWG 2005, Porto de Galinhas, Brazil, pp 49–56, 25–29 Sept 2005

Jacky J (1997) The Way of Z: practical programming with formal methods. Cambridge University Press

Limbourg Q, Vanderdonckt J, Michotte B, Bouillon L, López-Jaquero V (2004) UsiXML: A language supporting multi-path development of user interfaces. In: Proceedings of 9th IFIP working conference on engineering for human-computer interaction jointly with 11th international workshop on design, specification, and verification of interactive systems, EHCI-DSVIS'2004, Kluwer Academic Press, pp 200–220

Loer K, Harrison MD (2002) Towards usable and relevant model checking techniques for the analysis of dependable interactive systems. In: Emmerich W, Wile D (eds) Proceedings 17th international conference on automated software engineering, IEEE Computer Society, pp 223–226. http://citeseer.ist.psu.edu/loer02towards.html

Michotte B, Vanderdonckt J (2008) Grafixml, a multi-target user interface builder based on usixml. In: ICAS '08: Proceedings of the fourth international conference on autonomic and autonomous systems (ICAS'08), IEEE Computer Society, Washington, DC, USA, pp 15–22. doi:10.1109/ICAS.2008.29

Oliveira R, Dupuy-Chessa S, Calvary G (2015) Plasticity of user interfaces: formal verification of consistency. In: Proceedings of the 7th ACM SIGCHI symposium on engineering interactive computing systems, EICS 2015, Duisburg, Germany, June 23–26, 2015, pp 260–265

Paternò FM, Sciacchitano MS, Lowgren J (1995) A user interface evaluation mapping physical user actions to task-driven formal specification. In: Design, Specificationa nd Verification of Interactive Systems. Springer, pp 155–173

Puerta A, Eisenstein J (2002) XIML: a universal language for user interfaces. In: Intelligent user interfaces (IUI). ACM Press, San Francisco

Roseman M, Greenberg S (1996) Building real-time groupware with groupkit, a groupware toolkit. ACM Trans Comput-Hum Interact 3(1):66–106

Thimbleby H (2015) Safer user interfaces: a case study in improving number entry. IEEE Trans Softw Eng. doi:10.1109/TSE.2014.2383396