Introduction

XIAOJUN BI,
ANDREW HOWES,
PER OLA KRISTENSSON,
ANTTI OULASVIRTA,
JOHN WILLIAMSON

This book is concerned with the design of interactive technology for human use. It promotes an approach, called computational interaction, that focuses on the use of algorithms and mathematical models to explain and enhance interaction. Computational interaction involves, for example, research that seeks to formally represent a design space in order to understand its structure and identify solutions with desirable properties. It involves building evaluative models that can estimate the expected value of a design either for a designer, or for a system that continuously adapts its user interface accordingly. It involves constructing computational models that can predict, explain, and even shape user behaviour.

While interaction may be approached from a user or system perspective, all examples of computational interaction share a commitment to defining computational models that gain insight into the nature and processes of interaction itself. These models can then be used to drive design and decision making. Here, computational interaction draws on a long tradition of research on human interaction with technology applying human factors engineering (Fisher, 1993; Hollnagel and Woods, 2005; Sanders and McCormick, 1987; Wickens et al., 2015), cognitive modelling (Payne and Howes, 2013; Anderson, 2014; Kieras and Hornof, 2017; Card, Newell, and Moran 1983; Gray and Boehm-Davis, 2000; Newell, 1994; Kieras, Wood, and Meyer, 1997; Pirolli and Card, 1999), artificial intelligence and machine learning (Sutton and Barto, 1998; Brusilovsky and Millan, 2007; Fisher, 1993; Horvitz et al., 1998; Picard, 1997; Shahriari et al., 2016), information theory (Fitts and Peterson, 1964; Seow, 2005; Zhai, 2004), design optimization (Light and Anderson, 1993; Eisenstein, Vanderdonckt, and Puerta, 2001; Gajos and Weld, 2004; Zhai, Hunter and Smith, 2002), formal methods (Thimbleby, 2010; Dix, 1991; Harrison and Thimbleby, 1990; Navarre et al., 2009), and control theory (Craik, 1947; Kleinman, Baron, and Levison, 1970; Jagacinski and Flach, 2003; Sheridan and Ferrell, 1974).

Computational interaction is a science of the artificial (Simon, 1996), where the object of research is the construction of artefacts for stated goals and function. Herbert Simon took up construction as the distinctive feature of disciplines like medicine, computer science, and engineering, distinguishing them from natural sciences, where the subject is natural phenomena, and from arts, which may not share interest in attaining goals: 'Engineering, medicine, business, architecture, and painting are concerned not with the necessary but with the contingent, not with how things are but with how they might be, in short, with design' (Simon, 1996, p. xiii).

Simon made three important observations, which we share, about sciences of the artificial. First, he distinguished the 'inner environment' of the artefact, such as an algorithm controlling a display, and the 'outer environment', in which the artefact serves its purpose. When we design a user interface, we are designing how the outer and the inner environments are mediated. Both the human and the technology also have invariant properties that together with the designed variant properties shape the process and outcomes of interaction (see Figure I.1). The final artefact reflects a designer's implicit theory of this interplay (Carroll and Campbell, 1989). When we design computational interaction, we are designing or adapting an inner environment of the artefact such that it can proactively, and appropriately, relate its functioning to the user and her context. In computational interaction, the theory or model, is explicit and expressed in code or mathematics. Second, Simon elevated simulation as a prime method of construction. Simulations, or models in general, allow 'imitating' reality in order to predict future behaviour. Simulations support the discovery of consequences of premises which would be difficult, often impossible, to obtain with intuition. In computational interaction, models are used offline to study the conditions of interaction, and they may be parameterized in a real-time system with data to infer user's intentions and adapt interaction. Third, Simon pointed out that since the end-products are artefacts that are situated in some 'outer environments', they can and should be subjected to

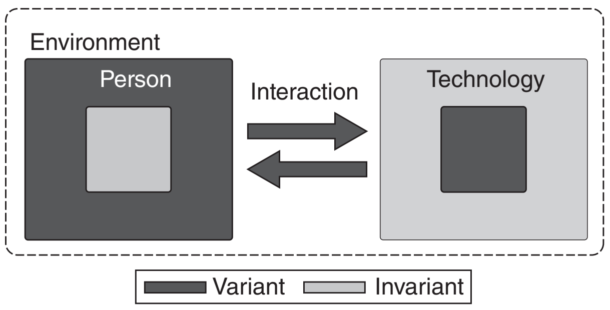

Figure I.1 Interaction is an emergent consequence of both variant and invariant aspects of technology and people. Technology invariants include, for example, current operating systems that provide an ecosystem for design. Technology variants include programmed interfaces that give rise to apps, visualizations, gestural control, etc. A person's invariants include biologically constrained abilities, such as the acuity function of human vision. A person's variants consist of the adaptive behavioural strategies. To design interaction is to explain and enhance the variant aspects given the invariants (both human and technology). In computational interaction, variant aspects are changed through appropriate algorithmic methods and models.

empirical research. Design requires rigorous empirical validation of not only the artefact, but also of the models and theories that created it. Since models contain verifiable information about the world, they have face validity which is exposed through empirical observations. They do not exist as mere sources of inspiration for designers, but systematically associate variables of interest to events of interaction.

However, computational interaction is not intended to replace designers but to complement and boost the very human and essential activity of interaction design, which involves activities such as creativity, sensemaking, reflection, critical thinking, and problem solving (Cross, 2011). We hold that even artistic and creative efforts can greatly benefit from a well-articulated understanding of interaction and mastery of phenomena captured via mathematical methods. By advancing the scientific study of interactive computing artefacts, computational interaction can create new opportunities and capabilities in creative and artistic endeavors. We anticipate that the approaches described in this book are useful in the interaction design processes of ideating, sketching, and evaluating, for example. Computational interaction can also be studied and developed in broader contexts of socio-technical systems, where power, labour, and historical settings shape interaction.

The fundamental ideas of computational interaction have been present for many years. However, there is now strong motivation to collect, rationalize, and extend them. In the early days of interaction design, design spaces were relatively simple; input devices, for example, were explicitly designed to be simple mappings of physical action to digital state. However, computer systems are now vastly more complex. In the last two decades, interaction has broken out from the limited domains of workstations for office workers to pervade every aspect of human activity. In the current mobile and post-PC computing era, new technologies have emerged that bring new challenges, for which the traditional hand-tuned approach to design is not well equipped. For example, wearable computing, augmented and virtual reality, and customizable interactive devices, pose increasingly wicked challenges, where designers must consider a multiplicity of problems from low-level hardware, through software, all the way to human factors. In these increasingly complex design spaces, computational abstraction and algorithmic solutions are likely to become vital.

Increasing design complexity motivates the search for a scalable interaction engineering that can complement and exceed the capabilities of human designers. A contention of this book is that a combination of methods from modelling, automation, optimization, and machine learning can offer a path to address these challenges. An algorithmic approach to interaction offers the hope of scalability; an opportunity to make sense of complex data and systematically and efficiently derive designs from overwhelmingly complex design spaces. This requires precise expression of interaction problems in a form amenable to computational analysis.

This book is part of an argument that, embedded in an iterative design process, computational interaction design has the potential to complement human strengths and provide a means to generate inspiring and elegant designs. Computational interaction does not exclude the messy, complicated, and uncertain behaviour of humans. Neither does it seek to reduce users to mechanistic caricatures for ease of analysis. Instead, a computational approach recognizes that there are many aspects of interaction that can be augmented by an algorithmic approach, even if algorithms are fallible at times and based on approximations of human life. Furthermore, computational interaction has the potential to reduce the design errors that plague current interactive devices, some of which can be life ending. It can dramatically reduce the iterations cycles required to create high-quality interactions that other approaches might write off as impossible to use. It can specialize and tailor general interaction techniques such as search for niche tasks, for instance, patent searches and systematic reviews. It can expand the design space and enable people to interact more naturally with complex machines and artificial intelligence.

In parallel with the increased complexity of the interaction problem, there have been rapid advances in computational techniques. At the fundamental level, rapid strides in machine learning, data science, and associated disciplines have transformed the problem space that can be attacked, generating algorithms and models of transformative power. At the practical level, affordable high-performance computing, networked systems, and availability of off-the-shelf libraries, datasets, and software for computational methods, make it feasible to analyse and enhance the challenging problems of interaction design. These are developments that should not be ignored.

This book sets out a vision of human use of interactive technology empowered by computation. It promotes an analytical, algorithmic, and model-led approach that seeks to exploit the rapid expansion in computational techniques to deal with the diversification and sophistication of the interaction design space. As noted by John Carroll (1997), and by others many times after, research on human-computer interaction has had problems assimilating the variety of methodologies, theories, problems, and people. The implementation of theoretical ideas in code could become a substrate and a nexus for computer scientists, behavioural and social scientists, and designers alike. If this promise is to be fulfilled, then serious effort must be made to connect mainstream search in human-computer interaction to computational foundations.

I.1 Definition

Defining a field is fraught with dangers. Yet there are two reasons to attempt one here: (i) to provide a set of core objectives and ideas for like-minded researchers in otherwise disparate fields to coalesce; (ii) to help those not yet using computational methods to identify the attributes that enable those methods to be successful and understand how this way of thinking can be brought to bear in enhancing their work. While any definition is necessarily incomplete, there is a strong sense that there is a nucleus of distinctive and exciting ideas that distinguish computational from traditional human-computer interaction research. The goal of our definition is to articulate that nucleus.

Definition Computational interaction applies computational thinking—that is, abstraction, automation, and analysis—to explain and enhance the interaction between a user (or users) and a system. It is underpinned by modelling which admits formal reasoning and involves at least one of the following:

- a way of updating a model with observed data from users;
- an algorithmic element that, using a model, can directly synthesize or adapt the design;

- a way of automating and instrumenting the modelling and design process;
- the ability to simulate or synthesize elements of the expected user-system behaviour.

For example, the design of a control panel layout might involve first proposing an abstract representation of the design space and an objective function (for example, visual search performance and selection time) for choosing between variants. It might then involve using an optimization method to search the space of designs and analysing the properties of the proposed design using formal methods. Alternatively, explaining observed user behaviour given a specific design might be explained by first proposing abstract representations of the information processing capacities of the mind (for example, models of human memory for interference and forgetting), then building computer models that automate the calculation of the implications of these capacities for each particular variant across the distribution of possible users, and then analysing the results and comparing with human data.

As in engineering and computer science, the hallmark of computational interaction is mathematical—and often executable—modelling that connects with data. However, while computational interaction is founded on fundamental theoretical considerations, it is constructive in its aims rather than descriptive. It calls for empirical rigour in evaluation and model construction, but focuses on using computationally powered models to do what could not be done before, rather than describing what has been done before. To this end, it emphasizes generating constructive conceptual foundations and robust, replicable and durable methods that go beyond point sampling of the interaction space.

I.2 Vision

The overarching objective of computational interaction is to increase our capacity to reliably achieve interactions with desirable qualities. A generic capacity is preferable over point solutions, because we need to deal with diverse users, contexts, and technologies. By seeking shared, transferable solution principles, the long-term aim shall be to effect the runaway 'ratcheting' of a body of research that builds up constructively and compositionally; something that research on human-computer interaction has struggled to achieve in the past.

This motivates several defining qualities of computational interaction: verifiable, mathematical theory that allow results to generalize; scalable, theory-led engineering that does not require empirical testing of every variant; transparency and reproducibility in research; and the concomitant requirement for reusable computational concepts, algorithms, data sets, challenges, and code. In machine learning, for example, the pivot to open-source libraries, large state-of-the-art benchmark datasets, and rapid publication cycles has facilitated both the uptake of developments and progress rate of research.

This vision opens up several possibilities to improve the way we understand, design, and use interactive technology.

Increase efficiency, robustness, and enjoyability of interaction Models of interaction, informed by often large-scale datasets, can enable the design of better interactions. For instance, interactions can demand less time; reduce the number of errors made; decrease frustration; increase satisfaction, and so on. In particular, computational approaches can help quantify these abstract goals, infer these qualities from observable data, and create mechanisms through which interfaces can be optimized to maximize these qualities. A computational approach aims to create an algorithmic path from observed user-system interaction data to quantifiably improved interaction.

Proofs and guarantees A clear benefit of some approaches, such as formal, probabilistic, and optimization methods is that they can offer guarantees and even proofs for some aspects of solution quality. This may be valuable for designers who seek to convince customers, or for adaptive user interfaces, that can more reliably achieve desired effects.

Develop user-centred design Computational interaction not only supports but can re-envision user-centred design, where techniques such as parametric interfaces, data-driven optimization, and machine-learned input recognition create direct data paths from usage and observations to design and interaction. Computation allows interfaces to be precisely tailored for users, contexts, and devices. Structure and content can be learned from observation, potentially on a mass scale, rather than dictated in advance.

Reduce the empirical burden Models can predict much of expected user-system behaviour. Interaction problems can be defined formally, which increases our ability to reason and avoid blind experimentation. Computational methods should reduce the reliance on empirical studies and focus experimental results on validating designs based on sound models. Strong theoretical bases should move 'shot-in-the-dark' point sampling of designs to informed and data efficient experimental work.

Reduce design time of interfaces Automation, data, and models can supplant hand-tweaking in the design of interfaces. It should be quicker, and less expensive, to engineer complex interactions if the minutiae of design decisions can be delegated to algorithmic approaches.

Free up designers to be creative In the same vein, algorithmic design can support designers in tedious tasks. From tuning pose recognizers at public installations to choosing colour schemes for medical instrument panels, computational thinking can let designers focus on the big picture creative decisions and synthesize well-designed interactions.

Harness new technologies more quickly Hardware engineering is rapidly expanding the space of input devices and displays for user interaction. However, traditional research on human-computer interaction, reliant on traditional design-evaluate cycles, struggles to tackle these challenges quickly. Parametric interfaces, predictive, simulatable models, crowdsourcing, and machine learning offer the potential to dramatically reduce the time it takes to bring effective interactions to new engineering developments.

I.3 Elements

Computational interaction draws on a long history of modelling and algorithmic construction in engineering, computer science, and the behavioural sciences. Despite the apparent diversity of these disciplines, each provides a body of work that supports explaining and enhancing complex interaction problems. They do so by providing an abstract, often mathematical, basis for representing interaction problems, algorithms for automating the calculation of predictions, and formal methods for analysing and reasoning about the implications of the results.

These models differ in the way they represent elements of interaction: that is, how the properties, events, processes, and relationships of the human and technology are formally expressed and reasoned with. They may represent the structure of design, human behaviour, or interaction, and they may represent some tendencies or values in those spaces. They allow the computer to reason and test alternative hypotheses with the model, to 'speculate' with possibilities, and to go beyond observations. This can be achieved analytically, such as when finding the minima of a function. However, in most cases, the models are complex and permit no analytical solution, and an algorithmic solution is needed, in which case methods like optimization or probabilistic reasoning can be used. Such computations may give rise to predictions and interpretations that are emergent in the sense that they are not obvious given lower-level data. Another property shared by these approaches is that the involved models can be parameterized by some observations (data) that represent the problem at hand.

We have collected a few key concepts of interaction and design, which can be identified behind the many formalisms presented in this book. This is *not* meant to exhaustively list all concepts related to computational interaction and we invite the reader to identify, define, and elaborate additional concepts central to computational interaction, such as dialogues, dynamics, game theory, biomechanics, and many others.

Information Information theory, drawing from Shannon's communication theory, models interaction as transmission of information, or messages, between the user and the computer. Transmission occurs as selection of a message from a set of possible messages and transferring over a degraded (noisy, delayed, intermittent) channel. The rate of successful transmission, and the complexity of the messages that are being passed, is a criterion for better interaction. For instance, to use a keyboard, the user communicates to the computer system via the human neuromuscular system. A user can be seen as a source communicating messages in some message space over a noisy channel in order to transmit information from the user's brain into the computer system. Importantly, throughput and similar information theoretical constructs can be taken as objectives for design, modelled using human performance models such as Fitt's law and the Hick–Hyman law, and operationalized in code to solve problems. Applications include input methods, interaction techniques, and sensing systems.

Probability The Bayesian approach to interaction is explicit in its representation of uncertainty, represented by probability distributions over random variables. Under this viewpoint, interaction can be seen, for example, as the problem of inferring intention via evidence observed from sensors. Intention is considered to be an unobserved random variable existing in a user's mind, about which the system maintains and updates a probabilistic belief. An effective interaction is one that causes the belief distribution to converge to the user's true intention. This approach allows informative priors to be specified across potential actions and rigorously combined with observations. Uncertainty about intention can be propagated through the interface and combined with utility functions to engage functionality using decision-theoretic principles. As a consequence of proper representation of uncertainty, Bayesian approaches offer benefits in terms of robustness in interaction.

One major advantage of the probabilistic view is that many components at multiple levels of interaction can be integrated on a sound basis, because probability theory serves as a unifying framework; for example, linking probabilistic gesture recognizers to probabilistic text entry systems. Tools such as recursive Bayesian filters and Gaussian process regression can be used directly in inferring user intention.

Learning Machine learning equips a computer with the ability to use data to make increasingly better predictions. It can transform static data into executable functionality, typically by optimizing parameters of a model given a set of observations to minimize some loss. In computational interaction, machine learning is used to both predict likely user behaviour and to learn mappings from sensing to estimated intentional states. Supervised machine learning formulates the problem as learning a function $y=f(x)$ that transforms observed feature vectors to an output space for which some training set of observations are available. After a training phase, predictions of y can be made for unseen x. Estimation of continuous values such as, for instance, intended cursor position, is regression, and estimation of discrete values such as, for example, distinct gesture classes, is classification. Supervised machine learning can replace hand-tweaking of parameters with data-driven modelling. There are many high-performing and versatile tools for supervised learning, including support vector machines, deep neural networks, random forests, and many others. Unsupervised learning learns structure from data without a set of matching target values. Techniques such as manifold learning (learning a simple smooth low-dimensional space that explains complex observations) and clustering (inferring a set of discrete classes) have potential in exploring and eliciting interactions. Unsupervised learning is widely used in recommender systems and user-modelling in general, often with an assumption that users fall into distinct clusters of behaviour and characteristics.

Optimization Optimization refers to the process of obtaining the best solution for a defined problem. For example, a design task can be modelled as a combination of elementary decisions, such as which functionality to include, which widget type to use, where to place an element, and so on. Optimization can also use constraints to rule out infeasible designs. Several approaches exist to modelling design objectives, ranging from heuristics to detailed behavioural, neural, cognitive, or biomechanical models. The benefit of formulating a design problem like this is that powerful solution methods can be exploited to find best designs automatically, rooted on decades of research on optimization algorithms for both offline and online setups.

States State machines are a powerful formalism for representing states and transitions within an interface. This model of interaction captures the discrete elements of interfaces and represents states (and their groupings) and events that cause transitions between states. Formalisms such as finite-state machines (FSMs), and specification languages such as statecharts, allow for precise analysis of the internal configurations of interfaces. Explicit modelling permits rigorous analysis of the interface properties, such as reachability of functionality, critical paths, bottlenecks, and unnecessary steps. Graph properties of FSMs can be used to model or optimize interfaces; for example, outdegree can be used to study the number of discrete controls required for an interface. State machines offer both constructive approaches in rigorously designing and synthesizing interfaces and in analysing and characterizing existing interfaces to obtain quantifiable metrics of usability.

Control With roots in cybernetics and control engineering, control theory provides a powerful formalism for reasoning about continuous systems. In applications to human-technology interaction, the user is modelled as a controller aiming to change a control signal to a desired level (the reference) by updating its behaviour according to feedback about the system state. The design of the system affects how well the user can achieve the goal given its own characteristics. Control theory views interaction as continuous, although a computer may register user behaviour as a discrete event. Modelling interaction as a control system with a feedback loop overcomes a fundamental limitation of stimulus–response based approaches, which disregard feedback. The control paradigm permits multi-level analysis, tracing the progression of user-system behaviour over time (as a process) to explain eventual outcomes to user. It allows insight into consequences of changing properties of the user interface, or the user.

Rationality Rational analysis is a theory of human decision-making originating in behavioural economics and psychology of decision-making. The assumption is that people strive to maximize utility in their behaviour. Bounded rationality is the idea that rational behaviour is constrained by capacity and resource limitations. When interacting, users pursue goals or utility functions to the best of their capability within constraints posed by user interfaces, environments, and tasks. The idea of bounded rationality is explored in information foraging theory and economic models of search.

Agents Bounded agents are models of users that take action and optimally adapt their behaviour to given constraints: environments and capabilities. The bounds include not only those posed by the environment, which includes the interface, but limitations on the observation and cognitive functions and on the actions of the agent. These bounds define a space of possible policies. The hypothesis is that interactive behaviour is rationally adapted to the ecological structure of interaction, cognitive and perceptual capacities, and the intrinsic objectives of the user. The interactive problem can be specified, for example, as a reinforcement learning problem, or a game, and behaviour emerges by finding the optimal behavioural policy or program to the utility maximization problem. The recent interest in computationally implemented agents is due to the benefit that, when compared with classic cognitive models, they require no predefined specification of the user's task solution, only the objectives. Increasingly powerful representations and solution methods have emerged for bounded agents in machine learning and artificial intelligence.

The chapters of this book present further details on the assumptions, implementation, applications, as well as limitations of these elementary concepts.

I.4 Outlook

The chapters in this book manifest intellectual progress in the study of computational principles of interaction, demonstrated in diverse and challenging applications areas such as input methods, interaction techniques, graphical user interfaces, information retrieval, information visualization, and graphic design. Much of this progress may have gone unnoticed in mainstream human-computer interaction because research has been published in disconnected fields. To coalesce efforts and expand the scope of computationally solvable interaction problems, an exciting vista opens up for future research.

Both the potential of and the greatest challenge in computational interaction lies in mathematics and algorithms. A shared objective for research is mathematical formulation of phenomena in human use of technology. Only by expanding formulations can one devise new methods and approaches for new problems. On the one hand, mathematics and algorithms are the most powerful known representations for capturing complexity. On the other hand, the complexity of a presentation must match the complexity of the behaviour it tries to capture and control. This means that the only way out of narrow applications is via increasingly complex models. The challenge is how to obtain and update them without losing control and interpretability.

A significant frontier, therefore, is to try to capture those aspects of human behavior and experience that are essential for good design. Whether users are satisfied with a design is determined not only by how much time they spend to accomplish the task, but also by its aesthetic aspect, the ease of learning, whether it fits the culture of certain regions, etc. Interaction is also often *coupled* with the environment and the situated contexts of the user. For example, a user's typing behaviour is dependent on whether a user is walking, if the user is encumbered, and the way the user is holding the device. The system itself is also often coupled to the environment. Also, computational interaction should not be restricted to model a single user's interaction with a single a computer system. A single user may interact with many devices, many users may interact with a single computer system, or many users may interact with many computer systems in order to for instance carry out a shared task objective. These challenges call for collaboration with behavioural and social scientists.

This increasing scale and complexity will pose a challenge also for algorithms. Algorithms underpin computational interaction and for systems implementing principles of computational interaction it is important that the underlying algorithms scale with increasing complexity. For example, naive optimization is often infeasible due to the complexity of the optimization problem. However, it is often relatively straight-forward to search for an approximately optimal solution. Complexity is multifaceted in computational interaction and may, for instance, concern the expressiveness of a system (for instance, the number of gestures recognized by the system), the ability of a solution to satisfy multi-objective criteria, or manage a complex and constantly changing environment. To increase our ability to deal with complex real-world phenomena implies that we need to search for more efficient ways to update models with data. Some computational models are heavily reliant on representative training data. A challenge in data collection is to ensure the data is accurately reflecting realistic interaction contexts, which may be dynamic and constantly changing, and that it captures the variability of different user groups. Moreover, data may not be available until the particular user has started adopting the design. Such a 'chicken and the egg' dilemma has long been a problem in computational interaction: the interaction data is needed for designing interface or interactive systems; yet the data will not be available until the design or system is available and the user starts adopting it. These challenges call for collaboration with computer and computational scientists.

Finally, there's the human. Computational interaction can be viewed as a continuation of the long history of automation, which has undergone a series of victories and setbacks due to complexity causing the loss of agency, deskilling, demotivation, and so on. Once a computational interaction technique is operating there is a risk the user is losing agency as a consequence of an overly complex system aiding or overriding the user. Such problems are exacerbated when the system fails to correctly infer the user's intention, in particular, if the system fails in an unexpected way, or if it fails to offer suitable controls and interpretability. Computational interaction should offer appropriate degree of transparency that allows users to at understand the mechanisms leading to a particular system prediction or suggestion at such a level that they can achieve their goals. To do so effectively either requires understanding users' existing workflows and practices, users to adapt to new ways of interaction tailed for computational interaction design, or a combination of both. From the perspective of algorithms, even the design problem is centered around mathematics: the central problem for user interface design for algorithmic systems is to assist users in understanding and shaping control, learning, and optimization functions and guiding a system-informed exploration of the decision space. What is the best user interface to mathematics? When this problem is successfully solved, computers can support users' creativity by assisting them in effectively exploring high-dimensional decision spaces. By modelling a domain-specific creative process, it is possible to optimize the creative process itself and help identify suitable solutions that are better according to some criteria, such as speed, aesthetics, novelty, etc. These challenges call for collaboration with designers and design researchers.

REFERENCES

Anderson, J. R., 2014. *Rules of the mind*. Hove, UK: Psychology Press.

Brusilovsky, P., and Millan, E., 2007. User models for adaptive hypermedia and adaptive educational systems. In: P. Brusilovsky, A. Kobsa, and W. Nejdl, eds. *The Adaptive Web: Methods and Strategies of Web Personalization*. Berlin: Springer, pp. 3–53.

Card, S. K., Newell, A., and Moran, T. P., 1983. The psychology of human-computer interaction. New York, NY: Lawrence Erlbaum.

Carroll, J. M., 1997. Human-computer interaction: psychology as a science of design. *Annual Review of Psychology*, 48(1), pp. 61–83.

Carroll, J. M., and Campbell, R. L., 1989. Artifacts as psychological theories: The case of human-computer interaction. *Behaviour and Information Technology*, 8, pp. 247–56.

Craik, K. J. W., 1947. Theory of the human operator in control systems: 1. the operator as an engineering system. *British Journal of Psychology General Section*, 38(2), pp. 56–61.

Cross, N., 2011. *Design thinking: Understanding how designers think and work*. Oxford: Berg.

Dix, A. J., 1991. Formal methods for interactive systems. Volume 16. London: Academic Press.

Eisenstein, J., Vanderdonckt, J., and Puerta, A., 2001. Applying model-based techniques to the development of UIs for mobile computers. In: *Proceedings of the 6th International Conference on Intelligent User Interfaces*. New York, NY: ACM, pp. 69–76.

Fisher, D. L., 1993. Optimal performance engineering: Good, better, best. *Human Factors*, 35(1), pp. 115–39.

Fitts, P. M., and Peterson, J. R., 1964. Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2), pp. 103–12.

Gajos, K., and Weld, D. S., 2004. SUPPLE: automatically generating user interfaces. In: *Proceedings of the 9th International Conference on Intelligent User Interfaces*. New York, NY: ACM, pp. 93–100.

Gray, W. D., and Boehm-Davis, D. A., 2000. Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied*, 6(4), pp. 322–35.

Harrison, M., and Thimbleby, H., eds., 1990. *Formal methods in human-computer interaction*. Volume 2. Cambridge: Cambridge University Press.

Hollnagel, E., and Woods, D. D., 2005. *Joint cognitive systems: Foundations of cognitive systems engineering*. Columbus, OH: CRC Press.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K., 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann, pp. 256–65.

Jagacinski, R. J., and Flach, J. M., 2003. *Control Theory for Humans: Quantitative approaches to modeling performance*. Mahwah, NJ: Lawrence Erlbaum.

Kieras, D. E., and Hornof, A., 2017. Cognitive architecture enables comprehensive predictive models of visual search. *Behavioral and Brain Sciences*, 40. DOI: https://doi.org/10.1017/S0140525X16000121

Kieras, D. E., Wood, S. D., and Meyer, D. E., 1997. Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 4(3), pp. 230–75.

Kleinman, D. L., Baron, S., and Levison, W. H., 1970. An optimal control model of human response part I: Theory and validation. *Automatica* 6(3), pp. 357–69.

Light, L., and Anderson, P., 1993. Designing better keyboards via simulated annealing. *AI Expert*, 8(9). Available at: http://scholarworks.rit.edu/article/727/.

Navarre, D., Palanque, P., Ladry, J. F., and Barboni, E., 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4). <doi:10.1145/1614390.1614393>.

Newell, A., 1994. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Payne, S. J., and Howes, A., 2013. Adaptive interaction: A utility maximization approach to understanding human interaction with technology. *Synthesis Lectures on Human-Centered Informatics*, 6(1): pp. 1–111.

Picard, R. W., 1997. *Affective Computing*. Volume 252. Cambridge, MA: MIT Press.

Pirolli, P., and Card, S. K., 1999. Information Foraging. *Psychological Review*, 106(4), pp. 643–75.

Sanders, M. S., and McCormick, E. J., 1987. *Human Factors in Engineering and Design*. Columbus, OH: McGraw-Hill.

Seow, S. C., 2005. Information theoretic models of HCI: a comparison of the Hick-Hyman law and Fitt's law. *Human-Computer Interaction*, 20(3), pp. 315–52.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N., 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1), pp. 148–75.

Sheridan, T. B., and Ferrell, W. R., 1974. *Man-machine Systems: Information, Control, and Decision Models of Human Performance*. Cambridge, MA: MIT Press.

Simon, H. A., 1996. *The Sciences of the Artificial*. Cambridge, MA: MIT press.

Sutton, R. S., and Barto, A. G., 1998. *Reinforcement learning: An introduction*. Volume 1, Number 1. Cambridge, MA: MIT Press.

Wickens, C. D., Hollands, J. G., Banbury, S., and Parasuraman, R., 2015. *Engineering Psychology & Human Performance*. Hove, UK: Psychology Press.

Thimbleby, H., 2010. *Press on: Principles of Interaction Programming*. Cambridge, MA: MIT Press.

Zhai, S., 2004. Characterizing computer input with Fitts' law parameters—the information and non-information aspects of pointing. *International Journal of Human-Computer Studies*, 61(6), pp. 791–809.

Zhai, S., Hunter, M., and Smith, B. A., 2002. Performance optimization of virtual keyboards. *Human–Computer Interaction*, 17(2–3), pp. 229–69.