Toward Usable Theories for Human-Automation Systems

Stéphane Chatty

Université de Toulouse- ENAC, 31055 Toulouse, France chatty@enac.fr

Abstract. A number of the engineering challenges raised by interactive software in the last decades have been addressed or worked around. When planning the development of large systems, user interfaces are often considered a solved problem and resources go elsewhere. However, it can be argued that some of the currently identified challenges (role of humans in large systems, explainability of AI) are reformulations of problems that are well known to the community of interactive systems engineering. We attempt here to distinguish which problems have been addressed by empirical methods and which remain to be solved by scientifical methods. We then propose requirements for theoretical models that will allow engineers to go beyond empiricism and increase their control over the systems they design and develop: engineers need models that consistently describe socio-cyber-physical systems, that are usable to address practical design questions, and that allow to automate part of the design and verification processes.

1 Introduction

For most engineers, user interaction is a solved problem. In the 1990s research in human-computer interaction has conceptualized various new technologies: augmented reality, virtual reality, ubiquitous computing, touch-based interaction, multimodal interaction, etc. In the 2000s it was unclear to the research community why these technologies were not more used industrially; a reasonable hypothesis was that engineering costs were too high and this was a good driver for research in the engineering of interactive computing systems. Then a succession of industrial successes have changed all assumptions: smartphones, tablets and web-based user interfaces have created vibrant ecosystems where hundreds of thousands of quality user interfaces are created on a regular basis. Designing and building user interfaces is a popular set of skills. R&D departments are turning to new challenges, mostly those raised by machine learning techniques.

Mission accomplished? Should the research community turn to new challenges as well? This article attempts to discern the remaining challenges behind the successes and to understand how the new engineering challenges raised by artificial intelligence converge with them. It advocates for a broader perspective on the engineering of human-computer systems, so as to propose solutions to industries who are facing the challenges of designing efficient and reliable human-automation systems.

With this perspective, the new challenges are higher than ever. They require the creation of a scientific body of knowlegde that systems engineers can use to describe and design hybrid systems made of humans, computers, and social systems such as rules and procedures.

2 HCI Engineering as a succession of challenges

Engineering disciplines are not self-driven: their goals are influenced by what the industry needs, and ultimately what society wants. In the case of human-computer interaction several technological waves can be identified, each corresponding to a new breakthrough of the HCI community and each revealing a new set of questions for engineering researchers.

Although it can be argued that job control languages were a user interface that is worth studying, the literature traces the first user interface engineering questions to command line interfaces. This is where questions of software architecture appeared: how can programmers organise their code in a manageable way, how can they manage the functional core and the user interface in relative independence?

Then came the largest wave, with the introduction of graphical displays. Not only did it reinforce the architecture challenges, it also raised questions of expressive power: how can a programmer or a researcher create a new interaction style such as iconic interaction, gesture interaction, etc? This set of challenges amplified as other interaction paradigms were introduced: information visualization, augmented reality, multimodality, etc.

The following wave came immediately behind the second. Now that the power of expression was growing, the expectations were growing too and came the serious engineering questions: what user interface for what use, what skills to design them, what design and engineering processes, how does one check that a design is satisfactory? These questions introduced a degree of confusion because very diverse phenomena and very diverse expertises are involved: software development issues, graphical issues, human behavior issues, design method issues. There still are relatively independent research communities who address engineering issues and sometimes reinvent solutions: human factors, programming languages, user interface design, user interface engineering, design thinking, and probably others.

More discreetely, the next wave was that of the safety-critical industries, and particularly aeronautics. Not only do their engineers have all of the above challenges, they also need to display a total control of the requirements of their systems and to implement a very demanding safety culture: one type of failure must never occur twice in the whole industry, and only unknown types of accidents are tolerated. The control of requirements is very demanding in terms of engineering processes, and the safety culture is very demanding in terms of scientific models so as to understand the causes of accidents and avoid their repetition.

3 The success of empiricism and brute force

Although no spectacular success has been recorded in engineering research, comparable to touch-based interaction and speech queries, the challenges raised by most of these successive waves have gradually faded. Most theoretical questions, including architecture, expressive power and model-based systems engineering are still not perfectly addressed. However, the progressive introduction of new constructions in programming languages have led to admissible practical solutions for architecture. The use of various programming patterns and Domain Specific Languages have made the expressivity

sufficient for a wide range of programmers to develop the interaction styles they desire. Participatory design methods and prototyping techniques have become widespread and have joined the wider movements of agile programming, design thinking and lean startup. Now even business schools teach methods than can be applied to interactive product design with reasonable success.

In practice, these successes are in large part those of empiricism and brute force. What the evolution of programming languages have permitted, particularly with Web technologies, is the recruitment of dozens of thousands of programmers with a shallow computing science background and still the abilty to produce useful code. This brute-force solution is perhaps not the success expected by researchers, nor the most efficient possible solution, but it is an effective solution. Similarly, the popularization of agile methods and design thinking derives from the implicit acceptance that no theory provides complete solutions for designing systems. Instead, every case is treated by gathering empirical data and iterating through approximative designs until finding an acceptable one.

Although very remote from design thinking, the current popularity of machine learning techniques is the culmination of this series of industrial progresses based on empiricism and brute force: software is expected to obtain acceptable results by itself through sheer statistics, without even having to gather more than mere examples.

4 From empiricism to theories

This combination of pragmatic approaches is an objective success: it is revolutionizing the whole field, including the engineering of safety-critical systems. But it is only a stage in the evolution of our field. Other engineering fields have gone through empirical phases before becoming science-based. For instance, architecture and civil engineering have been an empirical discipline long before structural analysis has become a modern science in the early 19th century. When this finally happened, creating buildings became more efficient in two ways: it required less building materials because smaller safety margins were required, and it required less manpower because computations required less skills.

This is where the remaining challenges from the previous technological waves combine with the new challenges coming with the new wave. They all highlight the need of better support for engineers who are responsible for the predictability of systems that combine humans and automation.

4.1 The remaining challenges

Safety-critical systems provide good examples of the remaining challenges. The hypothetical cause of the recent Boeing 737 MAX accidents is such an example. To start with, it is not satisfying that engineers can create conditions where users have no clue of what is happening, no knowledge of the existence of the malfunctioning subsystem, and no way of acting on it; it does not reflect full control of the whole human-computer systems by the engineers who design it. It is not satisfying either that certification experts were not able to detect the defective approach. The human-automation system that

they build and certify is apparently too complex for humans equipped with the available analytical frameworks. But what is worse is that comparable conditions had contributed to the AF447 accident ten years before. It seems that, in this field, the aeronautical sector is so ill equipped with conceptual frameworks that it does not learn efficiently from previous accidents.

Another example is the difficulties encountered in producing safety-critical software in air traffic control. Applying software assurance methods is slow, leaving enough time for changes in requirements to appear, thus requiring a new cycle of software assurance, and the whole process sometimes diverge.

4.2 The new technological wave

As said previously, most engineers consider that the user interface is a solved problem. There is little reason in ignoring this, even though we may have more efficient solutions in mind. Where the remaining issues are is where the behaviour of the whole *human-computer system* is studied, not only that of the user interface. Currently, this concerns the engineers of safety-critical systems who must eradicate so-called "human error"; tomorrow, it may be have a wider audience and even create opportunities to propose new solutions for interactive software itself.

With this perspective, machine-learning techniques are the new technological wave in human-automation systems. And the questions it raises are similar to the challenges above: being able to ensure that users will understand why an algorithm has reached a given state ("explainable AI") is the most cited issue, followed by the certifiability of AI-based software. In the future, being able to reason on human-automation systems may even become a long-term public issue; many of the questions asked today by the press to AI researchers about the consequences of AI on sociéty might be answered more efficiently by HCI engineers with the appropriate theoretical framework to reason on the behaviour of human-automation systems.

4.3 Getting rid of extra-theoretical analyses

All of these point at the deficiencies of the available theoretical frameworks, that leave a lot to extra-theoretical considerations. For instance, although it seems within our reach to represent the user's belief on the state of an interactive software in the same theoretical framework as the state itself, engineers still are condemned to using common sense for that. There are theories for software, theories for hardware, theories for human behaviour, but no common ground that allow to combine them.

Even when focusing on homogeneous system, notably software, there is another area where extra-theoretical considerations are required today: the verification of system properties. With few exceptions, the properties that can be checked within the bounds of theoretical systems are of very limited practical use: they do not correspond to the kind of questions that are asked by engineers, or the kind of invariants they need to check.

We propose the following common goal to all researchers in HCI engineering: *defining the theories that rid us of extra-theoretical reasoning when describing, designing, programming and verifying human-automation systems.* In our perspective, this is how model-based systems engineering will become of practical use to engineers.

5 Defining theories of human-computer systems

The challenges and the research program described above are far-reaching. Before being able to build formal theories, the research community will probably need to build ontologies that cover human-automation systems. Process-based ontologies are good candidates, but other options are probably as good. It is only equipped with a working ontology that candidate theories can be produced, tested and compared.

We propose here three requirements for such theories of human-automation systems. An acceptable theory must:

- provide consistent models for socio-cyber-physical systems; human behaviours, physical objects, and social rules must be interoperable and interchangeable within the theory.
- must be usable in practice to capture real designs and real design questions, including coding issues, so as to gain adoption from engineers and give them back control of their complex systems.
- must be usable to automate design processes and property verification, so as to gain adoption from industries, whose decisions are more and more based on immediate benefits.

In these requirements, "usability" can be understood in two meanings: the common sense meaning (the theory must "work"), but also the HCI meaning (they must be designed for ease of use by engineers). Not only should engineers be able to apply model-based engineering instead of extra-theoretical reasonings, they should also be able to do it while understanding at all steps what is happening. In the current situation, engineering often stop where 'human factors" start because engineers feel out of their depth in that area. But what the evolution of aviation safety teaches us is that this creates an area of improperly attributed responsibilities, that can lead to safety incidents, very slow engineering processes, or both.

6 Conclusion

The current successes of the digital industry force researchers on interactive systems engineering to reconsider their research directions. The heavy investments on machine learning techniques may even reduce available resources for their research. However, we have attempted to demonstrate here that there actually is a consistent set of engineering challenges in HCI engineering and AI engineering that point toward a common scientific problem: the creation of consistent and usable theories of human-automation systems, that can be used to describe and predict the behavior of these hybrid systems so that engineers can more effectively and efficiently exert control over them.

7 Acknowledgements

The author is director of the innovation programme at DSNA, 50 rue Henry Farman, Paris, France but this work is done as an associate researcher at ENAC. This work does not necessarily reflect the views of DSNA.