Suggestions for an EICS Roadmap

Anke Dittmar and Peter Forbrig
Dept. of Computer Science
University of Rostock
A.-Einstein-Str. 22
D-18051 Rostock, Germany
[anke.dittmar|peter.forbrig]@uni-rostock.de

ABSTRACT
The position paper considers three methodological challenges for Engineering Interactive Computing Systems (EICS): 1) better integration of design theories and practices from HCI and related fields into software engineering practices, 2) novel concepts to overcome limitations due to the separation of the user interface part and the application core of interactive systems, 3) advanced methods and tools for developing domain and user-specific interactive systems. It is suggested to create an EICS roadmap as result of the workshop.

ACM Classification Keywords
H.5 Information Interfaces and Presentation (e.g., HCI), H.1.2 User/Machine Systems, D.2.2 Design Tools and Techniques, D.2.10 Design, D.2.11 Software Architectures

INTRODUCTION

As the name would suggest, EICS is about providing methods, techniques, and tools to systematically develop interactive computing systems (ICS) of high quality. Yet, Ann Blandford asked in her keynote at EICS'2013 what engineering for interactive computing systems is. She pointed out that standard development practices for interactive systems such as iterative design are not particularly assigned to EICS and that the community needs to develop and maintain a better shared understanding of the nature, value and role of EICS to avoid becoming narrow and irrelevant [3].

'Traditionally', EICS approaches apply knowledge from computer science, software engineering (SE), and human-computer interaction (HCI) to design, implement, and reason about ICS and, in particular, user interfaces. Topics that are specifically addressed by EICS related conferences include ICS modeling, task-based and model-based design of user interfaces, formal methods for HCI, specification formalisms for interaction techniques, design spaces for organizing design parameters of advanced interaction techniques, and software architecture models and tools for designing, developing, and evaluating advanced user interfaces. In the workshop call, EICS is described as a "multidisciplinary endeavor positioned at the intersection of HCI, software engineering, interaction design, and other disciplines". These disciplines all contribute in one or another form to the "design, evaluation, and implementation of interactive computing systems for human use"[1] and consider themselves also as multidisciplinary. For example, SE is described as rooted in mathematics, computer science, engineering, natural sciences and humanities. Similar diagrams to the one in Figure 1 can be found in almost every schoolbook about interaction design or HCI. Each such diagram may be questioned in terms of mentioned influences and depicted intersections (e.g. Human Factors and HCI have no intersection in Figure 1).

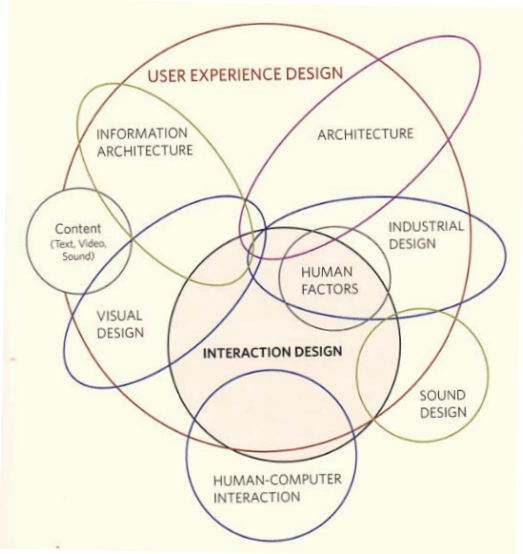

Figure 1: The disciplines surrounding interaction design (from [25]).

Figure 2 mentions, among many other disciplines, HCI and interaction design on the design side and SE on the technology side. EICS is not mentioned explicitly.

Before we further discuss the role of EICS and the expectations on ICS engineers, we would like to review the positioning of task analysis given in [5].

Excursus: Diaper's Understanding of Task Analysis for HCI

Dan Diaper considers in [5] HCI as "engineering discipline rather than science because its goals are inherently practical

[1] The quote is an essential part of the definition of HCI as discipline (see, e.g., http://hcibib.org/).

and involve satisfying design criteria." He suggests "that the historical division between HCI and software engineering is unfortune, as both study the same sort of systems for similar purposes." The difference between HCI and SE is "merely one of emphasis, with SE focusing more on software and HCI more on people" [5]. Diaper distinguishes between a narrow view on HCI focusing on the user-computer interface and a broad view on HCI considering "everything to do with people and computers" [5]. The latter view also includes the functionality of software systems because it affects the allocation of functions and the division of labor profoundly. Diaper thinks that task analysis (TA) is at the core of HCI and has to be better integrated into SE because software engineers and system analysts often do TA implicitly and poorly.

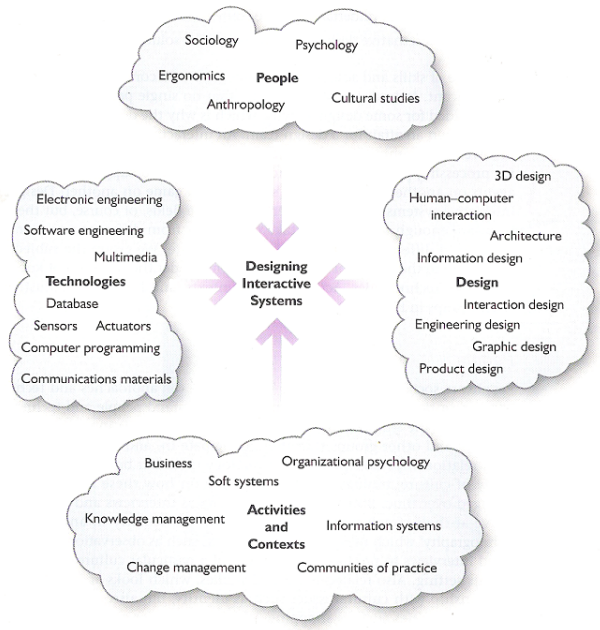

Figure 2: Disciplines Contributing to Interactive Systems Design (from [1]).

There may be few people who consider HCI as engineering discipline, and there may be people who do not see TA at the core of HCI. One may agree or disagree with Diaper's views, but he points out some important issues. First, when it comes to the development of ICS, a fusion of various research fields is needed. Second, while each discipline comes with its own practices, attitudes, and interrelations, their focus of research and some of the divisions have also to be explained historically. Third, ICS are more than their user-computer interfaces. And a last point should be mentioned here. For Diaper, design is "a goal-directed activity involving deliberate changes intended to improve the current world." Models of the current and of envisaged worlds are required in this process to develop and to implement ideas of change [5].

METHODOLOGICAL CHALLENGES FOR EICS

EICS should contribute to a more effective integration of SE approaches and of approaches from HCI, interaction design, and other fields. Engineering user interfaces or novel interaction techniques is one important area of EICS, but it only covers the above mentioned narrow view on HCI. EICS conference topics such as requirements engineering and software architectures for interactive systems, integrating interaction design into the software development process, and engineering user experience better reflect the broader view on HCI.

What should be expected from ICS engineers? They need to acquire profound background knowledge in HCI and related fields which they are able to apply in engineering user interfaces. They also must be able to convey HCI related problems to other software developers and SE related problems to interaction designers and HCI experts so that these problems can be tackled in a holistic way.

In the remainder of this position paper, we discuss a better integration of design theories and practices into SE practices and consider the role of external design representations in this process. In addition, two other methodological challenges are mentioned that could be part of an EICS roadmap.

– Novel concepts to overcome limitations due to the separation of the user interface part and the application core of interactive systems.

– Advanced methods and tools for developing domain and user-specific ICS.

Integration of Design Theories and Practices into Software Engineering Practices

Gould and Lewis are among the first exponents of user-centred design ideas. In a paper published 1985, they claim the need for an early focus on users and tasks, empirical measurement, iterative design and prototyping, and integrated usability design [15]. Until now these ideas are not fully integrated into SE practices. In [21], SE is characterised as "both a creative and a step-by-step process, often involving many people producing many different kinds of products". However, existing SE methods and recommended intermediate products of software projects reveal that the focus in SE is still to a large extent on functional aspects of software systems and on problem solving. Even requirements documents contain in most cases only the requirements on the software system under development, but rarely models of the current world or descriptions of other aspects of the envisaged world than the technical system (see the previous section).

Since the 1990ies, HCI puts more emphasis on design practices and theories (and interaction design developed as an own discipline). We are familiar with the main ideas of scenario-based design [24], participatory design, contextual design [2], and design rationale [19]. We know theoretical

frameworks and concepts such as distributed cognition [16] and situated action [27] and know about their consequences on design. The interplay between problem setting and problem solving and the role of external design representations are better understood [22,26,12]. However, SE practices are not fully integrated into the above mentioned design approaches. In [11], Dix et al. state, for example, that "the ideal model of iterative design, in which a rapid prototype is designed, evaluated and modified until the best possible design is achieved… is appealing" but that it is also important to be able to overcome bad initial design decisions or to understand the reasons behind usability problems and not just detect the symptoms. The authors recommend using iterative design "in conjunction with other, more principled approaches to interactive system design" (see a discussion in [8]).

We see one important role of EICS in bridging the gap between SE practices and design practices and theories from HCI and interaction design. Our own contributions are presented in [7,8,9,10]. For example, a lightweight use of formal methods is suggested in [9,10] to integrate evolutionary and exploratory prototyping of interactive systems in a systematic way. Evolutionary prototyping is especially recommended in SE when requirements of an application cannot be fully understood in advance [4].

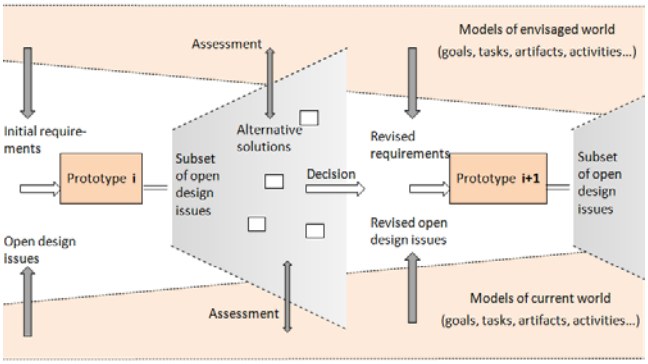

Figure 3: Overview of the model-guided prototyping approach suggested in [9,10].

Figure 3 illustrates the overall approach. The white arrows and the boxes indicate the evolution of the prototypical implementation over time. This prototype has to be deliberately underdesigned with respect to design issues where a clearer understanding of the problem and possible solutions needs to be obtained. In each iteration step, selected open design questions are explored by the development of alternative solutions to extend the evolutionary prototype. A technique called parallel model-guided prototyping is applied to develop these 'throw-away extensions' and to allow their assessment with both analytical and empirical means. Models of the current and envisaged world guide and constrain this process. In this way, an intertwined problem setting and problem solving is supported (see [9,10] for more details).

Co-Evolution of Different Design Representations

Although it is unquestioned by all contributing disciplines that in interactive systems design different kinds of design representations (models) are needed, their effective co-evolution and coupling remains problematic. In [23], Robinson and Bannon show effects of using representations of work (see also Figure 4). They point out that such models pass through different groups and are used for different purposes (ontological drift). While analysts create and use *descriptions of work* to understand their nature and to redesign it, software developers are interested in deriving ICS specifications from such models, which when implemented become *prescriptions for work* (flip-over effect).

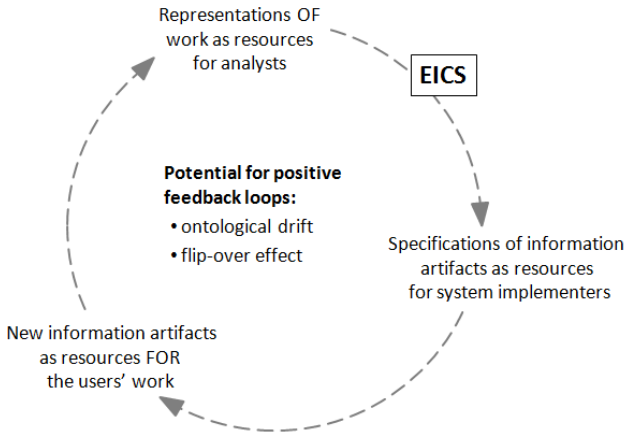

Figure 4: Effects of using representations of work in the design process of ICS [23].

The position of EICS is depicted in Figure 4. Models of users, tasks, context of use etc. are applied to create, refine, test, assess, and validate system specifications. In [6,7], we argue that task-based design approaches in EICS often gear task models towards system specifications. Resulting negative effects are discussed, e.g., in [23,27]. However, ICS engineers should be familiar with a broad range of design representation and their possible interpretations to be able to mediate between the different stakeholders and their interests, especially between software engineers and HCI experts, interaction designers, and users[2].

[2] General limitations of notations and models are well revealed in [17] quoting Ferguson: *„In a wonderful book about mechanical and structural engineering, Eugene Ferguson complains that many engineering disasters have happened because modern engineers have been taught to pay too much attention to calculation and formal analysis of structures and too little to the physical reality of the world of which those structures are a part… In software engineering… we do pay a great deal of attention to techniques that are essentially notational, leaving us – like the engineers whose education Ferguson is criticising – paying too little attention to the incalculable complexity of engineering practice in the real world. Requirements are in the real world, not*

ICS are More Than User Interfaces

Separating the user interface from the remainder of the application is now standard practice in developing interactive systems [18]. While many EICS approaches focus on user interface design (and challenges of distributed UIs, multimodal UIs, the growing variety of devices and interaction techniques… may have reinforced this trend), it is still a second-class issue in SE. Although a separation has many advantages, the development of the user interface and the functional core of an interactive system cannot be approached in a fully isolated way because certain usability concerns have to be considered already in the software architecture. Cancellation is a well-known example of an important usability feature which is often poorly supported in applications [18]. John et al. propose usability-supporting architectural patterns as a solution and as a means to educate software architects ([18], see also [13]). Such patterns describe the usability context (situation and potential usability benefits) and the problem (forces exerted by the environment and task, by human desires and capabilities, and by the state of the software system). In a pattern description, it is distinguished between a general solution in terms of general responsibilities that resolve above mentioned forces, and a specific solution that also takes into account the forces from prior overarching design decisions in a specific project context [18].

The separation of the user interface part and the application core is even more problematic for systems supporting a flexible allocation of functions, for adaptive systems, or for systems that are developed in an evolutionary way. Novel concepts to overcome limitations of this separation have to be developed.

Methods and Tools for Domain and User-Specific ICS

EICS-conference topics also include:

- Domain-specific languages for interactive systems,
- End-user development of interactive systems,
- User interface software and technologies for ambient assisted living,
- Engineering complex interactive systems (e.g., large datasets, large communities, enterprise systems).

This list indicates a third methodological, and perhaps also practical challenge. EICS approaches should demonstrate their applicability to specific domains and user groups.

EXPECTATIONS ON THE WORKSHOP

Roadmaps support the orientation of a field by giving an overview and highlighting open research issues. Good examples are presented in [14,20]. At the workshop, we would like to develop a shared view on an EICS roadmap. We think that it would be great to collaboratively create a 'roadmap-paper' after the workshop (which perhaps could be published in the EICS proceedings?).

CONCLUSIONS

The paper has shown that diverse disciplines contribute to the design of interactive computing systems. It has particularly discussed the role EICS can play in bridging SE and HCI (and related fields). EICS will be successful if it becomes irrelevant or a true sub-field of HCI and/or SE.

REFERENCES

1. Benyon, D., Turner, P., and Turner, S. Designing interactive systems: people, activities, contexts, technologies. Addison-Wesley (2005).
2. Beyer, H., Holtzblatt, K.: Contextual Design – Defining Customer-Centered Systems. Morgan Kaufmann Publishers (1998).
3. Blandford, A. Engineering works: what is (and is not) engineering for interactive computer systems? In *Proc. EICS '13*. ACM (2013), 285-286.
4. Davis, A.M. Operational Prototyping: A New Development Approach. *IEEE Softw. 9(5)*:70–78 (1992).
5. Diaper, D. Understanding Task Analysis for Human-Computer Interaction. In: *Diaper, D., Stanton, N.A. (eds.): The handbook of task analysis for human-computer interaction.* Lawrence Erlbaum Associates (2004).
6. Dittmar, A., Forbrig, P.: Task-based design revisited. In *Proc. of EICS '09* (2009). 111-116.
7. Dittmar, A., Harrison, M.D. Representations for an iterative resource-based design approach. In: *Proc.* EICS '10 (2010), 135-144.
8. Dittmar, A., and Forbrig, P. Selective Modeling to Support Task Migratability of Interactive Artifacts. In *INTERACT (3), vol. 6948 of LNCS*, Springer (2011), 571–588.
9. Dittmar, A. and Piehler, S. A constructive approach for design space exploration. In *Proc. EICS '13*. ACM (2103), 49-58.
10. Dittmar, A., and Schachtschneider, R. Lightweight Interaction Modeling in Evolutionary Prototyping. In *Proc. of FMIS workshop at EICS'13* (2013).
11. Dix, A., Finlay, J.E., Abowd, G.D., Beale, B. Human-Computer Interaction (3rd Edition). Prentice-Hall (2003).
12. Dix, A., and Gongora, L. Externalisation and design. In *Proc. of DESIRE '11*, ACM (2011), 31–42.
13. 'Engineering for HCI: Upfront effort, downstream pay-back': http://www.youtube.com/watch?v=gxiA4JTS9P8, at CHI 2013.
14. Garlan, D. Software architecture: a roadmap. In *Proc. of ICSE '00* (2000), 91-101.

---

in the machine. We must focus on them directly, and describe them conscientiously".

15. Gould, J.D., Lewis, C. Designing for usability: key principles and what users think. *Communications of the ACM 28(3),* (1985), 300-311.
16. Hollan, J., Hutchins, E., and Kirsh, D. 2000. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Trans. Comput.-Hum. Interact.* 7, 2 (Jun. 2000), 174-196.
17. Jackson, M.: A Discipline of Description. In *Proc. CEIRE98, Special Issue of Requirements Engineering*, Vol. 3(2) (1998), 73-78.
18. John, B.E., Bass, L., Sanchez-Segura, M-I., and Adams, R.J. Bringing usability concerns to the design of software architecture. In *Proc. EHCI-DSVIS'04*, Springer (2004), 1-19.
19. Moran, T., and Carroll, J. (eds.) Design Rationale: Concepts, Techniques, and Use, Lawrence Erlbaum Associates, Inc. (1996).
20. Nuseibeh, B., and Easterbrook, S. Requirements engineering: a roadmap. In *Proc. of ICSE'00*. ACM (2000), 35-46.
21. Pfleeger, S.L. Software Engineering: Theory and Practice, 2nd Edition, Prentice Hall (2001).
22. Rittel, H. W. J., and Webber, M. M. Dilemmas in a General Theory of Planning. *Policy Sciences 4* (1973), 155–169.
23. Robinson, M., and Bannon, L. Questioning representtations. In *Proc. of the ECSCW'91* (1991), 219-233.
24. Rosson, M.B., and Carroll, J.M. Usability Engineering – Scenario-Based Development of Human-Computer Interaction. Morgan Kaufmann Publishers (2002)
25. Saffer, D. Designing for Interaction. New Riders (2009).
26. Schön, D. The reflective practitioner: How professionals think in action. New York, Basic Books (1983).
27. Suchman, L. Plans and Situated Actions: The Problem of Human Machine Interaction. Cambridge University Press (1987).