HCI Engineering

Charting the Way towards Methods and Tools for Advanced Interactive Systems

EICS 2014 workshop

June 17, 2014
Rome, Italy

Jürgen Ziegler, José C. Campos and Laurence Nigay
(Organizers)

IFIP working group 2.7/13.4 on User Interface Engineering

Contents

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).

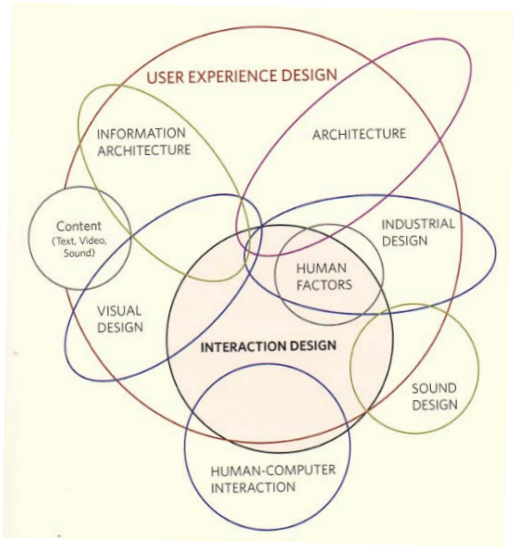


Figure 1: The disciplines surrounding interaction design (from [25]).

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.

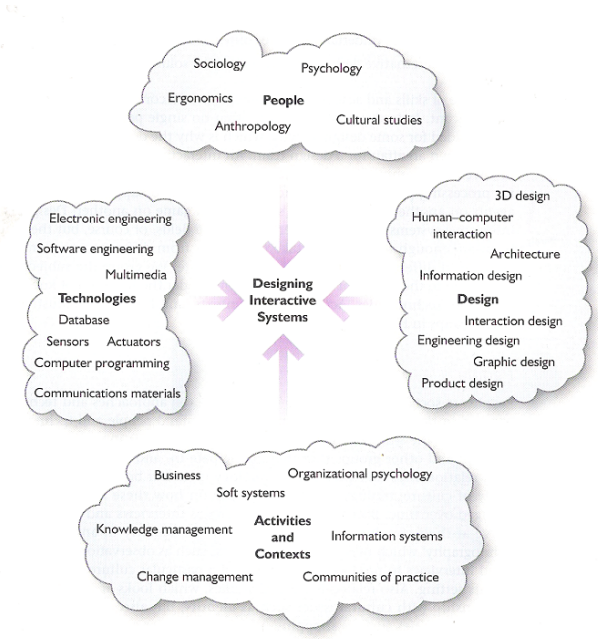


Figure 2: Disciplines Contributing to Interactive Systems Design (from [1]).

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].

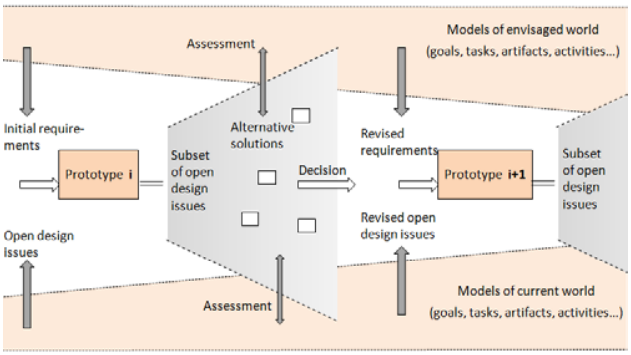


Figure 3: Overview of the model-guided prototyping approach suggested in [9,10].

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).

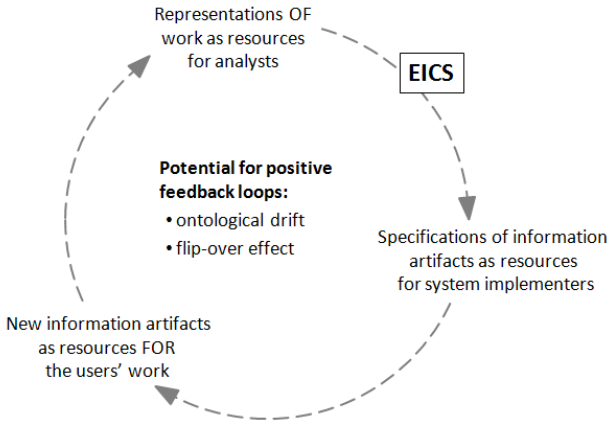


Figure 4: Effects of using representations of work in the design process of ICS [23].

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).

Quality: a challenge for advanced user interfaces

Sophie Dupuy-Chessa
Univ. Grenoble Alpes, LIG
CNRS
41 rue des mathématiques
38400 Saint Martin d'Hères, France
Sophie.Dupuy@imag.fr

Gaëlle Calvary
Univ. Grenoble Alpes, LIG
CNRS
41 rue des mathématiques
38400 Saint Martin d'Hères, France
Gaelle.Calvary@imag.fr

ABSTRACT
In Human Computer Interaction, universal quality does not exist. Despite all the design efforts, there will always be users and situations the user interface (UI) will not be suitable for. This is particularly true for advanced UIs for which quality criteria are still ill-defined. This paper addresses the engineering of UIs from the end-user's point of view: it does not address the internal quality such as the software architecture. It reviews ways for integrating quality all over the development process and from different points of view: 1) the quality intended by designers thanks to flexibility and creativity in the design process as well as verification; 2) the quality perceived by end-users thanks to UI adaptation and self-explanation.

Keywords
Quality, Development process, Design time, Run time.

PROBLEM: THE MULTI FACES OF QUALITY

End-users often find problems while interacting with advanced user interfaces (UIs). Questions about where is an option on the mobile phone version of an application, what is the gesture to accomplish a task, or why did something happen in the UI naturally arise due to the imperfect quality of the UI.

This problem of insufficient quality can be due to the increasing difficulty of designing advanced UIs by adding parameters like the devices, the location, the user characteristics... As mentioned by [12], the difficulty to design systems has been moved up. Even if the designer intends to achieve a good quality level (intended quality), he/she cannot foresee all these problems and obstacles at design time because each single user has his/her own understanding of the UI and might be in specific situations. It becomes impossible to provide support for all the users at design time for all the situations they might be in.

But quality problems also exist because as the user is not the designer, the user has a different understanding of the UI. He/she can encounter different problems or obstacles during the interaction process, which can make him/her perceive a bad quality level.

These problems are graphically represented by a gulf (figure 1) between the intended versus perceived quality.

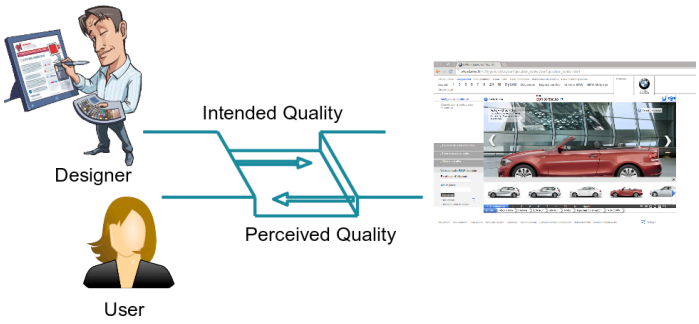


Fig. 1- Gulf of quality between intended and perceived quality.

Moreover perceived and intended quality is not static and can evolve with people, the context of use and the system itself. So the research question is:
What can be an engineering approach to continuously improve quality of advanced UIs?

MODELS: A UNIFYING APPROACH

As it is often the case in an engineering approach, we propose to use models as the basis for engineering quality. Models can serve as unifying approach that bridge the gap between the designer's and the end user's point of view:

- Models are created by developers along the development process. They aim at achieving the intended quality.
- Models created by developers can be reused to increase the quality perceived by end-users. They are "hyphens" between the intended and perceived qualities.

RESEARCH AXES: INTENDED AND PERCEIVED QUALITY

Considering the gulf of quality, we propose to structure the research axes thanks to designers' and end users' points of view. We need to study the intended and perceived quality.

Intended quality

Intended quality reflects the quality that the designer would like to achieve. It is mainly related to design

time where design, but also quality verification activities usually take place.

Quality by design: model-based approaches have been investigated for long for generating UIs from models. However, the resulting quality was rather limited: the UIs were simply made of basic widgets (e.g., input fields, radio buttons), far from supporting the advanced features promoted in ambient intelligence. Recently, **creativity** has been explored by models [10]: the point is no more to generate UIs for end-users, but UIs for developers as means for supporting the divergence and convergence processes in creativity [9]. The UIs are generated from a task model using different models transformations. The developer selects the UIs or parts of the UIs he/she really appreciates. Then genetic mutations are processed to make the models transformations evolve and thus give rise to new UIs. Figure 2 presents examples of UIs automatically produced for a Chat application. The UI variations (structure, widget, layout, colour, etc.) are intended to inspire the developer.

Fig. 2 – Examples of UIs generated by Magellan, a genetic algorithm-based environment for fostering developers' creativity

Another possible approach to improve the intended quality is to bring **flexibility for developers** so that to comply with the different practices in UIs development. Flexibility has been identified in the literature as one of the main research goals of method engineering [1]. For instance, [16] introduces flexibility in the design process for adaptive UI in order to decrease the threshold of use of models. We define three forms of flexibility:

1. Variability as the possibility of choosing one path in a set of equivalent variants. For example, instead of creating concrete user interfaces, they are generated from existing UIs, saving considerable part of the effort to be made for learning the CUI model and (re)modeling the UIs.
2. Granularability as the ability of a process model to support elements with different granularities, e.g. with different languages and/or quantities of details. We propose various levels of details when configuring and executing our tool for generating the domain model from a database. Expert designers just execute the tool whilst step by step explanations are provided for novice developers.
3. Completeness as the possibility of fulfilling or not the proposed process; some activities and/or artefacts are then optional or can be replaced by a predefined result or product. For instance, the activity "define the platforms model" can be avoided; in this case, the platform model can be replaced by "default" models that the developer picks up in a repository proposed by the process model.

Obviously, the UIs produced by such a flexible development process cannot be "perfect". However thanks to the process flexibility, designers and developers can reuse parts of their know-how and competencies, and are able to transfer some existing components into the paradigm of models: this makes it possible for them to create a first, albeit imperfect, version of their UIs, that they can iteratively improve, acquiring the needed competencies step by step.
But the solutions proposed are always limited; in particular they do not consider the enactment of the process, which is primordial when considering the need of rapid evolution when designing advanced UIs.

Quality by verification: Design should consider the verification of the intended quality. High quality of user interfaces, which can be ensured by several ways. For example, [13] proposes four ways of evaluation: *formally* by some analysis techniques, *automatically* by a computerized procedure, *empirically* by experiments with users and *heuristically* by simply looking at the UI and passing judgement according to one's own opinion. The automation of quality verification has been largely studied in the UI literature, however the usual techniques for UIs verification such as model checking [3] or testing [4] must be adapted to advanced UIs.

Perceived quality

Perceived quality corresponds to the end users' point of view under the system quality. So it is related to the runtime understanding of the system. At runtime, we consider that quality can be improved by composing existing UIs, by adapting UIs to their context of use or by explaining UIs.

Quality by reuse: Software composition is said to be

one of the grand challenges for the coming years. In the engineering of human computer interaction, this means being capable of composing UIs from existing pieces of software. It has been addressed for different software development paradigms including models [11]. The problem is to succeed in composing without impairing the UI quality. As a matter of fact, composition can introduce some inconsistencies or discontinuities in the final UI.

Quality by adaptation: Plasticity refers to the capacity of UIs to withstand the variations of the context of use (user, platform, environment) while preserving user-centered properties [5]. User-centered properties clearly refer to the perceived quality. So to improve this perception, UIs can be technically adapted in two main manners: adaptation is either a remolding (e.g., replacing an image with a text) or a UI redistribution among the set of available platforms (e.g., migrating the inputs to a mobile device). Model driven adaptation has been intensively studied given rise to a reference framework [6]. To consider quality in this framework, usability is introduced: usability criteria complement transformations between models so as to choose an adaptation among others [14].

To implement this approach, [15] proposes UsiComp, an integrated and open framework which implements the principles of Cameleon by allowing designers to create models and to modify them at design time and at runtime. For instance, Figure 3 shows the two different UIs that are produced by UsiComp for two platforms, a PC and a mobile phone. In the background, we can see the UI adapted to the screen of the PC platform. Among others, the original screen from the PC platform has been split into two tabs due to the small resolution of the mobile phone screen. The zoom controller of the map widget has been removed as well. With such adaptations, UIs are adapted to devices thus providing usable UIs.

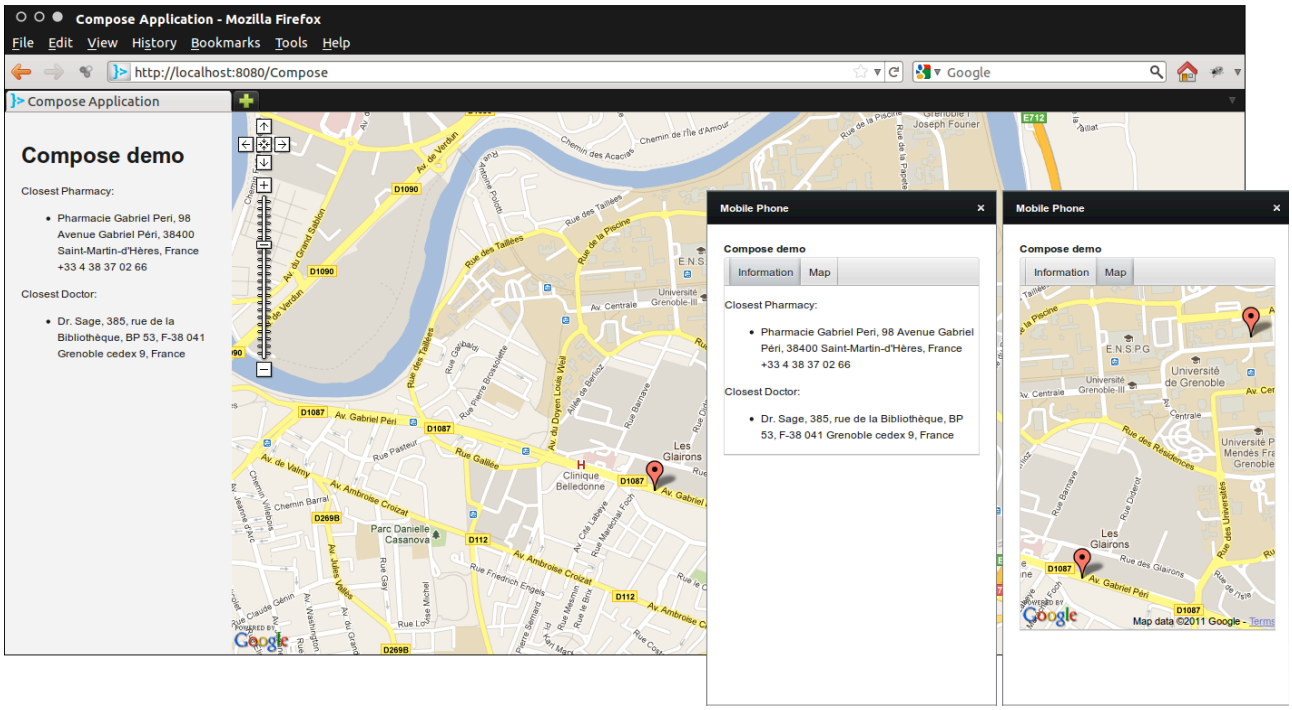


Figure 3. Generated UIs from the same task model. The UI in the background has been generated for a PC screen with higher resolution than the UI for the mobile phone in front of the figure.

However it still remains a challenge to guarantee that an adaptation has not impaired the perceived quality. Moreover adaptation generally does not take into account post-WIMP UIs.

Quality by repair: As universal quality is utopian, end-users are those who are the best to improve their UIs as soon as they grasp the purpose and design rationale of each UI element. So an interesting

approach is to support end-users reprogramming thanks to models [7]. This approach is not restricted to repair, it can also be used for design. In this case, it introduces new problems in quality by design as end-users become designers.

A complementary approach can propose to provide help about the UIs thanks to models: models created at design time can be used at runtime to explain UIs. In particular, these self explanatory UIs can provide the end-users with the design rationale of the UI [8].

For instance, [8] presents a system that consists of using the design models to compute questions and answers at runtime to provide an help system (Figure 4). The design models are still those proposed by the Cameleon reference framework.

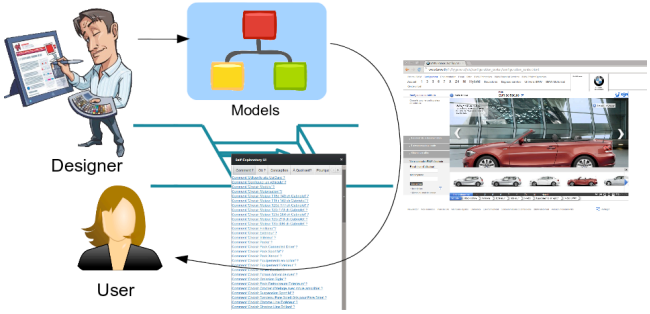


Figure 4. *Reducing the gap between intended quality and perceived quality by model-based explanations.*

The self-explanatory facilities generated with our approach are responsible for:

- Generating the set of questions. We consider those questions that the help system "knows" how to answer by inspecting the underlying models of the UI.
- Generating answers. Once the user asks a question to the help system, the system needs to compute an understandable explanation or answer. This is done through the following three steps:
    - Selecting the Explanation Strategy. In this phase the help system selects the explanation strategy according to the type of the question. For instance, a question about "how" to realize a task (e.g. how to choose packs when selecting a car) is associated to a strategy related to the task model.
    - Inspecting the models. Each explanation strategy inspects one or more models to retrieve the elements that have been defined for each strategy. For instance, to answer a "how" question, the strategy starts by inspecting the task model. The task related to the question is identified in the task model. Then the elements of the abstract UI mapped to the tasks are identified. Finally the elements of the concrete UI corresponding to the elements of the abstract UI are selected. Thanks to this chain of mapping, it is possible to obtain the final elements in the UI that can provide help. For instance, for the question "how to choose packs", the mapping between models allows the system to retrieve the « Packs » button in the UI.
    - Composing the answer. Once all the elements of the models have been retrieved, the answer is composed and prepared to be presented.
- Presenting the answer. The computed answer is provided to the user in an understandable way. For example, the system will propose to users to "Use the Packs button" if they want to choose a pack.

We conducted an experiment to evaluate the added value of model-based self-explanations. It shows that most of the users identify the help system as useful, in particular the How and the Where questions. However the study has also collected some interesting suggestions. First, we identified other types of questions (what is…, what if…) not explicitly supported by our system. Secondly usability of the help system, that was not our main concern, needs to be improved to facilitate interaction to select questions and to guide users thanks to the answer.

However they are limited to graphical UIs and there is a clear need to increase models so as to provide explanations about advanced UIs.

CONCLUSION

The multi-faces quality of advanced UIs requires continuous amelioration. This challenge motivates needs for co-evolution between actors and systems: roles are unified between end users and developers; the gap between design, run and evaluation times are erased so as to improve quality at any time.

REFERENCES

1. Harmsen F., Brinkkemper S., and Oei J., Situational method engineering for informational system project approaches, in *Methods and Associated Tools for the Information Systems Life Cycle*, 1994, p. 169‑194.

2. Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1990), 249–256.
3. Paternò, F., and Santoro, C. Support for reasoning about interactive systems through human-computer interaction designers' representations. *Comput. J. 46*, 4 (2003), 340–357.
4. Hjort, U. H., Illum, J., Larsen, K. G., Petersen, M. A., and Skou, A. Model-based gui testing using Uppaal at Novo Nordisk. In *FM 2009: Formal Methods*. Springer, 2009, 814–818
5. Thévenin D. and Coutaz J., Plasticity of user interfaces: Framework and research agenda ; in Proceedings of Interact'99, A. Sasse & C. Johnson (réds), IFIP IOS Press Publ., 1999, 110–117.
6. Calvary, G., Coutaz J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces, Interacting With Computers, Vol. 15/3, 2003, 289-308.
7. Sottet, J.-S., Calvary, G., Favre, J.-M., and Coutaz, J. Megamodeling and metamodel-driven engineering for plastic user interfaces: Mega-ui. In *Human-Centered Software Engineering*. 2009, 173–200.
8. Garcia Frey, A., Calvary, G., Dupuy-Chessa, S., and Mandran, N. (2013). Model-based self-explanatory UIs for free, but are they valuable? In *Proceedings of the 14th IFIP TC13 Conference on Human-Computer Interaction (INTERACT'13), 2-6 September 2013, Cape Town, South Africa*. Springer.
9. Buxton, B. Sketching User Experiences: Getting the Design Right and the Right Design, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007
10. Masson, D., Demeure, A., **Calvary**, G. Magellan, an Evolutionary System to Foster User Interface Design Creativity, Proceedings of the second ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010), ACM Press, Berlin, pp 87-92
11. Gabillon, Y., Petit, M., **Calvary**, G., Fiorino, H. Automated planning for User Interface Composition, 2nd SEMAIS Workshop, IUI conférence, Feb. 2011, Palo Alto, USA
12. Palen, L. Beyond the Handset: Designing for Wireless Communications usability, ACM Transactions on Computer-Human Interaction, 9(2), pp. 125-151, june 2002.
13. Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1990), 249–256.
14. Sottet, J.-S., Calvary, G., Coutaz, J., and Favre, J.-M. A model-driven engineering approach for the usability of plastic user interfaces. In *Proc. of EIS '07*, Springer-Verlag, 2008, 140–157.
15. Garcia-Frey A., Ceret E., Dupuy-Chessa S., Calvary G., and Gabillon Y. UsiComp: an extensible model- driven composer. EICS, (2012), 263–268.
16. Céret E., Calvary G., Dupuy-Chessa S., Flexibility in MDE for scaling up from simple applications to real case studies: illustration on a Nuclear Power Plant, 25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13, Bordeaux, France, 2013.

Individual Cognitive Abilities and Styles in HCI: Three Main Challenges and a Tiered Adaptation Model

Sven Bertel
Usability Research Group, Faculty of Media, Bauhaus-Universität Weimar
Bauhausstr. 11, 99423 Weimar, Germany sven.bertel@uni-weimar.de

ABSTRACT
For various tasks in human-computer interaction, measures of performance and emotion can be improved by adapting the user interface to a user's individual cognitive profile. Such tasks can be found, for example, with eLearning, information visualization, gaming, and human-computer collaboration in reasoning or problem solving (e.g. in design). Relevant factors within a cognitive user profile may include separate cognitive abilities, styles, and preferences, as well as personal characteristics of memory or attention. The three chief challenges for a successful adaptation to a user's cognitive profile lie (1) in establishing which of these factors are relevant for a given task-user pair, and to which extent, (2) in establishing how, based on (1), adaption may best be performed, and (3) in establishing a user's individual values for these factors. All three challenges possess aspects related to cognitive theory and user modeling, as well as quite practical aspects related to measuring a cognitive profile. This contribution will start out by, in turn, addressing research questions and methods related to the challenges. A tiered structure for cognitive user models will be subsequently sketched, through which adaptation can be based on parameters that are individualized to varying degrees, depending on how much is known about a user's individual cognitive profile.

Author Keywords
Interaction Technologies; Adaptation; Cognitive Factors.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Some people are better than other people when it comes to processing information given to them as pictures. Some people outperform others when working with texts. The hypothesis that there exist inter-individual differences in how people process information on visual and verbal dimensions has been the focus of research for decades. For learning, for example, this has led to rather well established theories on different cognitive *abilities* (i.e. relating to differences in what people are capable of doing), different cognitive *styles* (i.e. relating to differences of modality in how they process and represent information), and different learning *preferences* (i.e. relating to differences in how they prefer information to be presented to them; cf. [21]). While eLearning studies into employing a learner's individual visual/verbal learning style for adapting learning material have so far shown mixed effects of adaptation on performance, they have shown clear benefits on intrinsic learner motivation [7, 2]. Would it not thus be useful if an eLearning system knew about a user's individual cognitive abilities and styles, and if it used such knowledge to automatically adapt how learning material gets presented?

A second example: A study by Keehner [17, 18] on effects of conflicting visual/haptic cues on surgeons' spatial scene understanding during laparoscopic surgery demonstrated that such effects depend on the surgeons' individual spatial abilities. One may conclude that, for surgeons with low general spatial abilities, specific types of visual/haptic cue mismatches should be avoided, as these lead to a decreased surgery-related scene understanding. Would it not be useful if the laparoscopic system knew about a surgeon's individual low spatial abilities and used such knowledge to prevent certain cue mismatches from being presented?

A third example: For human reasoning with spatio-temporal information (e.g., about distances/directions between objects or places) the construction process of corresponding mental scene models has been found to be frequently subject to preferences (e.g. [19, 15]) in that, among all possible consistent models, some are constructed more frequently (as well as faster and easier) than others. Parsimony seems to play a role, as models that reflect simple and typical configurations seem to be preferred [12], which can be explained by simple and commonly used mental processes [8]. Also, cultural left/right biases may have an effect on preferences [15]. Let us assume that we have a task that involves some spatial configuration and that is either shared by a human and a computational system, or in which a human user is assisted by a computational system. Would it not be useful if the system knew about such general human cognitive preferences, and adapted its actions accordingly (e.g., by occasionally drawing the user's attention also to some non-preferred model)?

It seems feasible to ground some of the adaptive behavior required for effectively addressing these examples onto general and general-purpose models of cognition, such as ACT-R [1], and to also, at least partially, evaluate adaptive behavior based on these. The third example is likely to be amenable to such an approach. For more specific cognitive faculties, use of more specific models of cognition may be required (e.g., for visuo-spatial reasoning, [8]), or of models which address specific classes of interactive tasks or domains (e.g., in-vehicle

user interfaces [25], web-based navigation [11], or the design of built environments [6]). For purposes of either usability engineering or testing of adaptive user interfaces, a few approaches exist that incorporate user models along with interaction models, with the user models targeting general or specific user groups (for the latter, e.g. [24]). Programmable user models [28] are a good example of approaches which specifically try to bridge gaps of knowledge, methods, and philosophy of approach between the cognitive scientist and the interface designer, as they permit predictive interface evaluations based on implemented psychological theories. Even if an individual user's previous domain knowledge and the history or context of use can often be modeled, when it comes to inter-individual differences in cognitive processing, such as may be expressed by cognitive abilities or styles, such models address the average user of the targeted group. The models are specific with respect to task, context, expertise, or group, but are still general with respect to that group's cognitive profile.

It should seem obvious that adaptive user interfaces which will adequately address the first and the second example will need to draw on more individual parameters than can be derived from general models of cognition, even from task- or group-specific ones. For the individual user, these parameters may be assessed through various test instruments (for example, through tests of mental rotation or mental perspective taking performance for the respectively related, specific cognitive spatial abilities). While definitions of the terms *cognitive abilities*, *styles*, and *preferences* vary in the literature, I will, for the current purpose, simply rely on the definitions by Mayer and Massa [21] that were already provided in the first paragraph above. As far as other terminology is concerned, I will follow [23] and hold an *adaptive* system to be one that automatically modifies some of its characteristics to better fit a user's needs. To keep the discussion short enough for these few pages, the reader may simply envision such adaptive behavior to occur either on macro or micro levels (cf. the discussion by [20]) and either at design- or run-time.

Next, three chief challenges for user interface design connected to such an individualized approach will be outlined. The third section will sketch how a practical design compromise may look like that mediates between the desire (or even, need?) to know as much about a user's cognitive profile as possible and severe limitations in obtaining such knowledge. The contribution will conclude with a short discussion.

THREE MAIN CHALLENGES

As it turns out, the first two examples above are still both of a rather benevolent type for the interface designer interested in how cognitive factors (i.e., abilities, style or preferences) vary among his user base. This is because the cited studies all investigated relationships between certain tasks (learning, scene understanding) and specifically selected cognitive abilities or styles (visual/verbal, spatial). When starting with a given task and an individual user only, however, the first question is which of the currently known, separate cognitive factors may play a role in determining how that one user cognitively tackles the task. This question is already much harder to answer. It becomes harder yet when asking for specific effect sizes and directions, let alone when asking for possible interactions between the factors. Hunt [14] compared seven different cognitive skills; all of these may easily differ between users and may influence user performance. These were reading comprehension, vocabulary, grammar, mechanical reasoning, quantitative skills, mathematics achievement, and spatial reasoning. Some of these skills are more intercorrelated than others, with reading comprehension and spatial reasoning forming the maximally unrelated pair. While this list of skills serves to give a flavor of the variety one has to be able to deal with, it is by no means exhaustive.

The **first main, hard challenge** is thus to establish which factors are actually important for adaptive user interfaces for each given class of tasks, and which are not. Given that HCI specialists are usually no specialists in the cognitive sciences, and vice-versa, this challenge is best addressed interdisciplinarily. It must certainly be largely addressed incrementally. It seems that it will not be enough to simply identify relationships between cognitive factors on the one hand and measures of task performance or satisfaction on the other, but that one has to equally demonstrate that specific adaptation strategies formed on top of discovered inter-individual differences in users' cognitive profiles will be effective. Very likely, not all differences in profiles will be equally open to on operationalization for adaptive user interfaces.

A related, hard problem is created by the fact that many distributions of cognitive ability or style expressions are far from being uniform. Usually, some style expressions will be more frequently encountered than others within a targeted user group. For example, with cognitive learning styles, a majority of learners will be visual (74%), especially when sampling from populations in the natural sciences or engineering (e.g., [9, 2]). This may be of quite some importance for scenarios in eLearning or eTutoring. When a user interacts with an adaptive system, asking how often his cognitive profile is to be encountered within a user population will likely seem to be irrelevant to him. What likely will be relevant to him is that any adaptation which is to occur based on information about cognitive profiles will occur based on *his* specific profile. For systematically investigating interrelationships between cognitive factors and tasks, as well as for investigating how an adaptation should best occur based on a given user profile, frequencies of specific types of user profile are, however, far from being irrelevant. When a user type is encountered too infrequently, achieving reliable statistical comparisons between types may be difficult. As it seems to be no viable option to exclude those users with rare profile types from using an interface, or to at least decide to not try to provide them with adaptive interface behavior, other approaches need to found. A possible approach seems to lie in using mixed-method designs that combine conclusions based on quantitative observations for the frequent types with conclusion based on qualitative observations for the more infrequent types.

Should one be in the lucky position to have already determined which cognitive factors effectively influence a given user's performance on a specific task, and in which ways, the **second main challenge** lies in determining how adaptation

of the interface should best be performed. In the case of the eLearning example above, this may be comparably straightforward, as one may choose the representational format of learning material to correspond to a user's individual cognitive profile. However, even with the example, it seems unclear if this would always be the best strategy. Depending on the specific learning goals and the context, it may be more important to train learners to instead better cope with material in formats that do not well match their cognitive profiles (see e.g. [9] for a discussion of this point for inter-individual differences of students' learning styles). The question thus becomes one of either adapting *to* a profile, or *against* it. This is particularly interesting in gaming applications, when system adaptation can take the form of a computational player either adapting to keep a human player in the game for as long as possible (and to, e.g., maximize fun or engagement) or to adapt to become a maximally strong adversary (see e.g. [27] for an example and a discussion).

Choices of either adapting to or against a user's individual cognitive profile may be limited for purely practical reasons for all sorts of asymmetric tasks which are not fully specified. Such tasks can be frequently encountered, for example, in computer-assisted design: here, usually only a subset of design requirements can be formally described (often, the more technical requirements, such as e.g. with regard to thermal insulation), while other requirements remain the exclusive province of a human designer (often, those regarding an aesthetical or a more holistic evaluation of the design). The result is a setting of an asymmetric human-computer collaboration, with shared initiative, in which both parties need to rather act as partners that observe, adapt to, and, ideally, anticipate the other's actions in order to be jointly effective [3].

Finally, the **third chief challenge** lies in making sure that a user interface has enough information about an individual user available to allow for an adequately precise establishing of that user's relevant cognitive profile. From a practical point of view, this likely poses the hardest of the three challenges. While test instruments exist for many of the various cognitive abilities and styles, administering them is often a rather lengthy process that, what is even worse, is often tiring and/or boring for the tested user. The same holds true for many of the established instruments that assess individual characteristics of a user's memory and attention systems. The best ways of measuring a user's working memory capacity are performance measures, that is, one tests how much one can cram into the memory system. However, running at full load will not only quickly tire a user out for further tests of cognitive ability or style, but also for any main tests of potentially adaptive systems that the HCI professional will be chiefly interested in. It thus seems highly impracticable to measure factors related a user's cognitive profile separately and over and over again for each task, interface, or tool. There will be neither enough time, cognitive user (or researcher) resources, or user motivation for such an endeavor.

In addition, individual expressions of abilities or style will be only largely indicative of a user's general traits in cognitive processing, though not necessarily for life and in all situations. Measurements and the factors that they reflect are each expected to vary to different degrees depending on the task or context (e.g. for learning styles, see [2]). Depending on the task at hand, it may be necessary to dynamically assess additional information about the individual user, for instance, to infer information about his current mental state (e.g., regarding current foci of attention). Different measurements of psychophysical parameters may be used to further inform, individualize, and situate more general cognitive user models, such as through parameters derived from gaze (e.g. [10], [4]), EEG, or skin conductance, etc. Where some of the criticism above was directed against cognitive user models that were too general (i.e., not individualized enough), the problem here is that information obtained about a user's individual cognitive profile via the established test instruments may be too general (i.e., not situation- or context-dependent enough) to serve as the sole base from which to derive parameters governing adaptive interface behavior. For example, for tasks or games that involve problem solving, one can assume reliable information about a user's current strategies and foci to likely be at least as important for a generation of effective adaptation strategies as reliable information about the user's individual cognitive profile, especially when such profile information has been obtained independent of task or context.

TIERED ADAPTATION

I have suggested above that a viable course of action to deal with different frequencies of cognitive user profiles in a user population may lie in combining largely qualitative measures applied to the infrequent types with largely quantitative measures used for the more frequent types. This may imply that different information obtained about a user's individual profile may be reliable to different degrees, depending on the methods through which it was acquired. It may also mean that adaptive interface behavior may have to be more or less assertive: less, if it is based on less reliable bits of information, more if it solidly grounds in reliable data. Such a graduated approach fits well with a situation in which obtaining any specific information about a given user-task pair (i.e., information about relevant cognitive factors which is either individualized or situated, or, better, both) is nearly always costly, with prices being chiefly paid in currencies of user fatigue or motivation. We thus need an approach that facilitates the striking of a practical compromise between a user interface designer's wish to know as much about a user's cognitive profile as is possible and strong practical limitations in obtaining such knowledge.

The model proposed here is based on a related sketch for situations of joint human-computer spatial reasoning and problem solving suggested by [5]. It conceptually extends [5] and is not limited to applications of spatial reasoning or problem solving. The model (see Fig. 1 for an illustration) consists of three tiers of user-related data, in which information available about the cognitive profile of an individual user is gradually refined from bottom to top level. The model is tiered, as whenever more specific information is lacking, it may be substituted by less specific information, albeit at a price, as we will see. General cognitive factors are such as may be obtained, for instance, through general models of human cog-

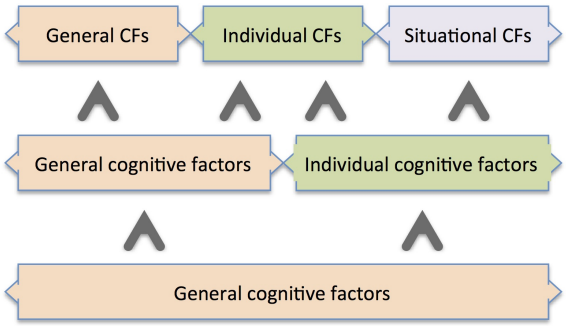


Figure 1: Sketch of the model with general, individual, and situational cognitive factors. Ideally, knowledge would exist up to the top tier for all factors relevant to adaption of a user interface behavior for a specific task. Where such is not available, less specific values may substituted from lower tiers.

nition, e.g. regarding general (i.e. averaged) characteristics of memory or attention, temporal characteristics of cognitive processing, or general preferences of mental model construction such as those described for the third example above. Individual cognitive factors of our modeled user are of the kind that can be obtained through test instruments, such as for various relevant cognitive abilities or styles. The first and the second of the examples discussed in the introduction would likely benefit from an adaptation rooted in general and individual cognitive factors. Situational cognitive factors, as shown on the top level, are such factors as can be derived based on live measurements of (e.g., various psychophysical) parameters about our user. With such measurements, one may attempt to address questions of the following kind: Which controls has the user gazed at over the course of the last five seconds? Is the user's current attention span likely to be lower or higher than his usual individual level? Are there any indications that his usual expressions of learning styles should be modified for the current task? etc. Situations related to the first two examples are easily conceivable during which the effectiveness of adaptive user interfaces may be improved by utilizing a selection of situational cognitive factors in addition to individual and general ones.

Through establishing the three levels, we get a number of interesting properties. First, information about factors on higher levels comes generally at higher costs than information about factors on lowers levels. Information obtained higher up is also more likely to be specific to a situation or task, and will be less well suited for drawing general conclusions about the cognitive profile of our user, or even about a group of users. General cognitive factors exist on all three levels, as only some of the more general assumptions and findings about a user's cognitive profile can and need be gradually refined along the way up. The same holds true for individual cognitive factors on levels two and three, and their relationship with situational cognitive factors. Last, and perhaps most importantly, we may use this model to sketch relationships between groups of factors which become useful when we do not have complete information about a user's individual cognitive profile (that is, virtually always). Ideally, our knowledge would extend up to the top tier for all of those factors that we judge to be relevant to effective adaption of a user interface for a specific task under consideration. Whenever such knowledge is incomplete, we may revert to using knowledge about factors on lower tiers, thus effectively using our model as one of graduated defaults. Such reversion to lower tiers will come at a price, of course, as we will lose some of our individualized or situated potential. In other words, whenever we move from higher to lower tiers, more specific values about cognitive factors and factor expressions will be substituted by less specific ones. It seems likely that one may often rather easily devise mechanisms of adaptive behavior that can reflect such changes in specificity, for example, by adapting a user interface less strongly (e.g., in less assertive and more subtle ways) whenever information about the individual user's cognitive properties is drawn from factors on less reliable tiers.

CONCLUSIONS

I have discussed three chief challenges that need be addressed to be able to productively harvest inter-individual differences in cognitive user profiles for generating effective, adaptive behavior of user interfaces. It seems that at least challenges (1) and (3) can only be adequately tackled inter-disciplinarily and through collaboration of user interface designers and cognitive scientists. In addition to incremental solutions for those two challenges, such collaboration should, at best, also result in a process of establishing a base of rules or best practices of which cognitive factors frequently relate to an individual user's task performance, and how, and of which test instruments are best to be used under which circumstances. The reason why I raise this point is that, naturally, we will very likely not see many user interface designers suddenly starting a training in the cognitive sciences. This would neither seem practicable, nor necessary. What is needed, however, is, first, an increased awareness among user interface designers for inter-individual differences rooted in their users' cognition and, second, approaches to designing adaptive user interfaces that permit scaling. This is to say that the approach needs to be able to scale from simple, recipe-like stages (à la "The ten most important rules for adapting your iOS app to your users' diversity in attentional resources", perhaps similarly simple and iconic as e.g. Shneiderman's *Golden Rules* [26] or Norman's *Design Principles* [22]) to much more detailed and focused stages in which specific cognitive factors will need to be addressed based on specific theories or (live) models of user's cognitive processes.

One should of course ask whether the three challenges that were raised here are the only challenges out there that currently keep us from creating effective adaptation of various user interfaces to the individual user's cognitive profile. The answer is no, of course. For the purposes of this contribution, I have tried to select and concentrate on the three challenges which I currently rate as the most urgent and difficult ones. Other, related challenges do, for instance, target questions of how changes of users' cognitive profiles over the course of a day, a task, or a lifetime, can be tracked, modeled, and reacted to, or of how users' cognitive abilities and styles interact with their emotional states. These are interesting questions, to be

sure; however, I would strongly recommend embarking on a stepwise process, in which we tackle the most important challenges first, before moving on.

As a last point, I would like to argue that we are currently seeing a significant increase in the frequency of HCI settings that would benefit from a better adaptation of the involved interfaces and systems to the individual user's cognitive properties. Let me illustrate this point through two quick examples: First, eLearning. The use of MOOCs (Massive Open Online Courses) is currently on a rapid upswing, no matter if counted by the number of courses being offered or by attendance (surpassing 230,000 individuals per course for some courses, [16]). Also, participants are drawn from increasingly heterogeneous groups [13]. One possibly effective response to diminishing available instructor resources per participant may lie in constructing eLearing systems that more closely shadow the individual user's learning progress than is currently the common case, and that better adapt to it, similarly to how a good tutor would adapt material and methods to a student's progress. Such response would certainly benefit in quality if designers of those eLearning systems would know more about users' individual cognitive profiles as well as know better how to adapt interface behavior to these.

The second example is based on an extrapolation about the frequency of human-computer interactive systems that are asymmetric in the sense sketched above. The more computational tools we see that employ processes which remain partly opaque to the standard user (often because of reasons of data or process complexity, e.g. in applications of computer-supported design or big data analysis, or that employ techniques of data mining or machine learning), the more frequently HCI researchers and practitioners will need to address issues of human-computer collaboration and negotiation in which adequate cognitive user models will be key for effective interaction. My bet is that, at least for as long as we will continue to see an increase in the use of data-intensive applications, we will see an increase of asymmetric interaction settings that can greatly benefit from effectively adapting to users' individual cognitive profiles.

REFERENCES

1. Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. An integrated theory of the mind. *Psychological review 111*, 4 (2004), 1036.
2. Beckmann, J., Bertel, S., and Zander, S. Adaptive elearning based on individual learning styles - performance and emotional factors. In *Proc. Mensch & Computer 2014* (to appear).
3. Bertel, S. Show me how you act on a diagram and I'll tell you what you think (or: Spatial structures as organizing schemes in collaborative human-computer reasoning). In *2005 AAAI Spring Symposium: Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance*, AAAI (2005), 77–82.
4. Bertel, S. Towards attention-guided human-computer collaborative reasoning for spatial configuration and design. In *Foundations of Augmented Cognition (Proceedings of HCI International 2007, Beijing)*, Springer; Berlin (2007), 337–345.
5. Bertel, S. Supporting joint human-computer reasoning and problem solving: Blueprint of an architecture. In *Proc. of Spatial Cognition 2010 (Posters)*, D. N. Rapp, Ed., SFB/TR 8 Report No. 024-07/2010 (2010), 6.
6. Bertel, S., Barkowsky, T., and Freksa, C. Mental model-centered design for built environments. In *Proc. of EDRAMOVE Workshop on Movement and Orientation in Built Environments, EDRA 39th Annual Conference*, SFB/TR 8 Report No. 015-05/2008 (2008), 13–19.
7. Brown, E., Brailsford, T., Fisher, T., Moore, A., and Ashman, H. Reappraising cognitive styles in adaptive web applications. In *Proceedings of the 15th international conference on World Wide Web*, ACM (2006), 327–335.
8. Engel, D., Bertel, S., and Barkowsky, T. Spatial principles in control of focus in reasoning with mental representations, images, and diagrams. In *Spatial Cognition IV. Reasoning, Action, and Interaction*, C. F. et al., Ed., LNCS, Springer (Berlin, 2005), 181–203.
9. Felder, R. M., and Brent, R. Understanding student differences. *Journal of Engineering Education 94*, 1 (2005), 57–72.
10. Freksa, C., and Bertel, S. Eye movements and smart technology. *Computers in biology and medicine 37*, 7 (2007), 983–988.
11. Fu, W.-T., and Pirolli, P. Snif-act: A cognitive model of user navigation on the world wide web. *Human–Computer Interaction 22*, 4 (2007), 355–412.
12. Goodwin, G. P., and Johnson-Laird, P. N. Reasoning about relations. *Psychological Review 112*, 2 (2005), 468–93.
13. Ha, T.-H. Moocs by the numbers: Where are we now? **`http://blog.ted.com/2014/01/29/moocs-by-the-numbers-where-are-we-now`**, 2014.
14. Hunt, E. B. Verbal ability. In *Human Abilities: An information-processing approach*, R. Sternberg, Ed. New York: W.H. Freeman & Co., 1985, 31–58.
15. Jahn, G., Knauff, M., and Johnson-Laird, P. N. Preferred mental models in reasoning about spatial relations. *Memory & Cognition 35*, 8 (2007), 2075–2087.
16. Jordan, K. Mooc completion rates: The data. **`http://www.katyjordan.com/MOOCproject.html`**, 2014.
17. Keehner, M. Conflicting cues from vision and touch can impair spatial task performance: Speculations on the role of spatial ability in reconciling frames of reference. In *Spatial Cognition VI. Learning, Reasoning, and Talking about Space*. Springer, 2008, 188–201.
18. Keehner, M. Spatial cognition through the keyhole: How studying a real-world domain can inform basic science—and vice versa. *Topics in Cognitive Science 3*, 4 (2011), 632–647.
19. Knauff, M., Rauh, R., and Schlieder, C. Preferred mental models in qualitative spatial reasoning: A cognitive assessment of allen's calculus. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, Erlbaum (1995), 200–205.
20. Leutner, D. Instructional-design principles for adaptivity in open learning environments. *Curriculum, plans and processes of instructional design: international perspectives* (2004), 289–307.
21. Mayer, R. E., and Massa, L. J. Three facets of visual and verbal learners: Cognitive ability, cognitive style, and learning preference. *Journal of educational psychology 95*, 4 (2003), 833.
22. Norman, D. A. *The design of everyday things*. Basic books, 2002.
23. Oppermann, R. *Adaptive user support: ergonomic design of manually and automatically adaptable software*. CRC Press, 1994.
24. Quade, M., Lehmann, G., Engelbrecht, K.-P., Roscher, D., and Albayrak, S. Automated usability evaluation of model-based adaptive user interfaces for users with special and specific needs by simulating user interaction. In *User Modeling and Adaptation for Daily Routines*, E. Martín, P. A. Haya, and R. M. Carro, Eds. Springer, 2013, 219–247.
25. Salvucci, D. D. Rapid prototyping and evaluation of in-vehicle interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI) 16*, 2 (2009), 9.
26. Shneiderman, B. *Designing the user interface: strategies for effective human-computer interaction*, vol. 2. Addison-Wesley Reading, MA, 1992.
27. Wetzel, S., Spiel, K., and Bertel, S. Dynamically adapting an AI game engine based on players' eye movements and strategies. In *Proc. of Sixth ACM SIGCHI EICS 2014* (to appear).
28. Young, R. M., Green, T., and Simon, T. Programmable user models for predictive evaluation of interface designs. *ACM SIGCHI Bulletin 20* (1989), 15–19.

Inside the 'Big Data' Factory: Lessons from Health Informatics

Jessica Wardlaw
University College London
Gower St, London, WC1E 6BT
j.wardlaw@ucl.ac.uk

Mordechai (Muki) Haklay
University College London
Gower St, London, WC1E 6BT
m.haklay@ucl.ac.uk

Anna L. Cox
University College London
Gower St, London, WC1E 6BT
anna.cox@ucl.ac.uk

ABSTRACT
This paper highlights wider engineering challenges encountered in the design and development of interfaces for 'Big Data' analysis tools with a small number of users in specialist application domains. Based on six years' collaboration with a UK-based health informatics company processing 16 million hospital admissions records a month, we report challenges associated with commercial pressures, rapid changes in the business environments, and variation in users' computer literacy, functional requirements, technical resources and knowledge of the complex underlying data. Whilst the literature advocates new interaction and analysis techniques for Big Data visualizations in these contexts, this paper cautions that some users cannot spend the time to learn how to interact with them or are constrained by their working environment. Despite these challenges, we show that User-Centred Design and research 'in the wild' can go some way to address these engineering challenges and support the implementation of Big Data business-to-business applications.

Author Keywords
Big data; health informatics; user-centred design

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

BIG DATA, BIG BUSINESS

The volume, velocity and variety of data being created and analysed is ever increasing [2]. IBM state that 90% of the world's data was created in the last two years and 2.5 million gigabytes of data is created daily – enough to fill 27,000 iPads per minute - a phenomenon known as 'Big Data'. Data has been compared to the "oil" of the 21st century [4], but, unlike oil, it will not run out. As mobile access to computers and the internet increases, new and emerging technologies will fuel the growth of data.

Big Data presents significant opportunities and is already transforming business sectors across the global economy: from increased transparency and accountability with open data, to new scientific discoveries, and market-changing products and services, which can be developed using modelling, data analytics and data-driven science. Its volume and variety, and the speed at which it is created and processed, however, can pose huge challenges.

Designing systems to cope with large volumes of data is not a trivial design problem, especially when the users do not have the requisite skills to analyse and visualise such large datasets. A shortage of skilled workers in the data analytics market is cited as one of the key barriers to further data analytics activity [2]; these challenges have hindered the realisation of the opportunities presented by Big Data [10]. User-Centred Design (UCD) is important in facing these challenges because consumers, businesses and academia require usable analytical tools to profit from Big Data; UCD can help to reverse this through the design of tools that effectively support workers to analyse and present an increasingly large volume and variety of data.

While much of the discussion surrounding Big Data applications considers large numbers of users, it is vital to understand the social context. In the 'Social Life of Information', Brown and Duguid remind us that, by engaging the social context in which technology is embedded, better designs and uses will emerge [1]. Technology is most successful where it augments the natural abilities of humans; for example, people identify others as individuals and interact with them in context in a way that computers are unable to replicate.

This position paper will further this idea by discussing the wider engineering challenges for the design and development of interfaces for Big Data analysis tools with a small number of users in specialist application domains. The lead author reflects on a six-year collaboration with Dr Foster Intelligence (DFI), a public-private partnership in between England's National Health Service (NHS) Information Centre and Dr Foster Ltd, who were pioneers in the UK health informatics sector. They provide independent health and social care information to healthcare managers and clinicians to improve clinical effectiveness and efficiency. DFI has produced a range of web-based data analysis tools which give NHS managers access to the Hospital Episodes Statistics (HES) database that contains admitted patient care data from 1989 onwards and outpatient attendance data from 2003 onwards. Whilst live access to 825 million hospital records presents many challenges, with users varying in computer literacy, functional requirements, technical resources and knowledge of the complex underlying data, it offers opportunities that are unavailable in any other country's health system.

Designing effective user interfaces for Big Data is a challenge that user-centred design and evaluation methods (UCM) from the field of Usability Engineering can overcome. The lead author collaborated with DFI to adapt UCM, which revealed the business realities of developing and marketing Big Data applications; they can be a high risk investment for small-medium enterprises. Competitive pressure to keep up with technological advances can tempt companies towards technology-led development processes.

This experience provides a unique perspective on the engineering-related challenges associated with the design of interfaces for Big Data applications and the methodological support required in a specific application domain. This will be informative for other enterprises looking to exploit the growing business-to-business market for Big Data.

INSIGHTS FROM INSIDE A BIG DATA BUBBLE

The major challenge for DFI is to process 16 million records a month, add value to them and enable users to run meaningful analysis; learners and novices need support to achieve their tasks and goals in a timely manner. Many different user types perform different roles (for example, Information Analysts, Clinicians, Public Health Analysts, Medical Directors) and correspondingly different tasks. Users therefore have a wide disciplinary background and vary in computer literacy, often correlated with how long they have been working in the NHS. (For example, users who have had a long career in the NHS can be accustomed to particular working practices and averse to change.) Furthermore, users are spread across England in hospitals that operate in subtly different ways. These user characteristics are not unique to health informatics and can be seen in other emerging Big Data application domains. These include financial services, insurance industry, local government, social services, urban planning, large infrastructure projects, environmental management and climatology, in which the end-users synthesize Big Data to make decisions that impact upon people's everyday lives.

Integrity of the data, transparency in the methodologies used and secure data processing are also important to DFI's customers. Whilst users require design and evaluation support to make sense of it and complete their tasks, they are knowledgeable about the highly complex information that the application processes. A Hospital Standardised Mortality Ratio (HSMR), and the methodology for its calculation, comprised the basis for the formation of the original Dr Foster Ltd, which hospitals use to benchmark their death rate; the organization also adds various flags to each admission record, including readmission within 30 days and patients' demographic and lifestyle characteristics. DFI store the data in an area of the office that requires security access and is protected computationally because of its personal and sensitive nature; the system displays an error message if the user runs a query with few results, which would potentially allow patients to be identified.

Finally there are more systemic, technological challenges. A screenshot study to inform personas development revealed that users integrate Big Data within their local working environment. The screenshot in Figure 1 shows Microsoft Word and Excel open simultaneously; this corroborated interview data that revealed many users rarely use anything other than Microsoft Office applications for their work. Users integrate the data and visualizations into Microsoft Word to create reports, Excel for analysis and PowerPoint to present at meetings. Such information must inform the design of the user interface and visualizations. They are familiar to certain icon and button conventions from these applications and find it difficult to adjust to others. The implications of this are that whilst the latest literature tends to endorse and advocate more current, novel and advanced visualizations [3,5,7], many users are not accustomed to the visualization techniques used and do not have the time to learn them. DFI's users are further limited by the technology available to them in a hospital. The technology infrastructure in NHS hospitals necessitates that tools developed by DFI are compatible with all web browsers; users rarely have a good network connection or the authority to download and use the latest browsers, or any additional software required to use some of the more advanced visual analytics tools. In subsequent interviews for the personas study, one user reported that they logged on in the evening with a glass of wine because their home network connection is better.

USER-CENTRED DESIGN OF METHODS AND TOOLS

A mixture of technical and organizational obstacles prevented the implementation of many recommendations that resulted from studies that the lead author carried out.

Commercial pressures, and the rapidly changing business environments in which Big Data applications are developed, create challenges for their development; many UCM demand more resources than commercial pressures permit Big Data enterprises to use. Whilst the authors'

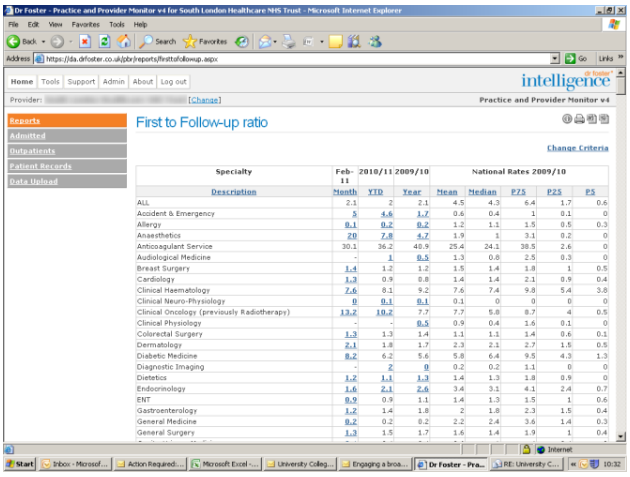


Figure 1 User-Generated Screen Capture

collaboration with DFI had a positive impact in promoting UCD within the organizational culture, the lead author was given limited contact with end users to carry out user testing for commercial reasons; the organization preferred that users did not see unfinished products so that they would not be discouraged to renew contracts and switch to the competition. User testing was eventually carried out in two phases late in the development process, when the organization was confident that the application was sufficiently functional and presentable; the resultant time pressure hampered recruitment of participants (six in phase one, nine in phase two) and implementation of their recommendations. The limited time available to participants, and their geographical spread, further delayed the scheduling of test sessions. Therefore enhanced 'Discount' UCM were used and adapted to the context [6].

Methods included user testing and an online questionnaire to explore the users' level of understanding of maps. Personas were also developed with database server log file analysis, a user-generated screen capture study and interviews during which participants demonstrated a typical task, to provide the developers with a better understanding of the users' goals, behaviours, attitudes, skills and working environment. The authors adapted the way they recruited participants, selected tasks, identified usability problems and reported results according to the business and project requirements [8]. This methodological approach offers unique insights into the challenges of developing usable Big Data applications and facilitates the co-design of new approaches to develop suitable interaction techniques.

Figure 2 The Office "Fishbowl"

Many widely-used UCM were not designed for complex systems such as Big Data applications and are therefore not necessarily appropriate. For the first phase of the collaboration the lead author managed the design and development of geographical analysis functionality for a tool for public health professionals. Geospatial tools have inherent difficulties for users that are not necessarily revealed using discounted UCM [9]. For users to exploit these datasets to their full potential therefore requires new approaches to their design and evaluation.

In addition, organizational and physical aspects of the work environment played an important role. For example, a glass wall (Figure 2) was required because of the data's sensitivity; only a few developers were permitted access to it. Developers likened working behind the wall to working inside a "fishbowl"; it created a 'silo' effect and impeded communication between developers and their colleagues, including designers and customer-facing teams.

CONCLUSION

This position paper demonstrates the need for design and interaction techniques for small-medium data-focused enterprises that develop specialist Big Data applications for a limited number of users who vary in computer literacy, functional requirements, technical resources and knowledge of the complex underlying data. A finite market of customers and high commercial stakes demands the development and adoption of ostensibly trivial and facile interaction techniques and UCD methods, which can be overlooked in the pursuit of more advanced, viscerally impressive, solutions. It also demonstrates a major engineering challenge facing the designers and developers of Big Data applications: many users do not have the time or technology to be able to learn and use more advanced visualizations. Although research 'in the wild' has elucidated difficulties of embedding UCD in such organizations, it can also support them to adopt UCD to their benefit and tackle the engineering challenges they face in the growing Big Data business-to-business environment.

ACKNOWLEDGMENTS

This work was undertaken for an Engineering Doctorate supported by the Engineering and Physical Sciences Research Council (EPSRC) and DFI, and Industrial Fellowship from the Royal Commission for the Exhibition of 1851. We also thank participants in the studies described.

REFERENCES

1. Brown, J.S. and Duguid, P. *The social life of information*. Harvard Business Press, Boston, MA, USA, 2002.
2. Department for Business, Innovation & Skills. *Seizing the data opportunity: A strategy for UK data capability*. London, UK, 2013.

3. Fisher, D., DeLine, R., Czerwinski, M., and Drucker, S. Interactions with big data analytics. *interactions 19*, 3 (2012), 50.

4. Gartner. Gartner Says Worldwide Enterprise IT Spending to Reach $2.7 Trillion in 2012. 2011. http://www.gartner.com/newsroom/id/1824919.

5. Grammel, L. How Information Visualization Novices Construct Visualizations. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 943–952.

6. Nielsen, J. Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. In R. Bias and D. Mayhew, eds., Academic Press, Boston, MA, USA, 1996, 245–272.

7. CACM Staff. Visualizations Make Big Data Meaningful. *Communications of the ACM (CACM) 57*, 6 (2014), 19–21.

8. Wardlaw, J., Haklay, M., and Cox, A.L. Adaptation of Method-resources Between Projects: A Case Study From a Dynamic and Complex Work Domain. *Proceedings of CHI 2013 Workshop "Made for Sharing: HCI Stories of Transfer, Triumph and Tragedy,"* (2013), 4–9.

9. Wardlaw, J. Principles of Interaction. In M. Haklay, ed., John Wiley & Sons Ltd., 2010, 179–198.

10. Wind-Cowie, M. and Lekhi, R. *The Data Dividend*. London, 2012.

Towards an engineering approach for advanced interaction techniques in 3D environments

Louis-Pierre Bergé, Emmanuel Dubois, Minica Houry-Panchetti, Mathieu Raynal, Cédric Sanza
University of Toulouse; IRIT; Elipse / Vortex
F – 31 062 Toulouse France
Louis-Pierre.Berge@irit.fr; Emmanuel.Dubois@irit.fr

ABSTRACT
In recent years, Virtual Environments have appeared in new areas such as mass-market, web or mobile situations. In parallel, advanced forms of interactions are emerging such as tactile, mixed, tangible or spatial user interfaces, promoting ease of learning and use. To contribute to the democratization of 3D Virtual Environments (3DVE) and their use by persons who are not experts in 3D and occasional users, simultaneously considering Computer Graphics and Human Computer Interaction design considerations is required. In this position paper, we first provide an overview of a new analytical framework for the design of advanced interaction techniques for 3D Virtual Environment. It consists in identifying links that support the interaction and connect user's tasks to be performed in a 3DVE with the targeted scene graph. We relate our work to existing modeling approaches and discuss about our expectations with regards to the engineering of advanced interaction technique.

Author Keywords
3D User Interaction, Design, Analytical Framework, 3D Interaction Techniques, Multimodal Interaction.

ACM Classification Keywords
D.2.2 [**Software engineering**]: Design Tools and Techniques - *User interfaces*; H.5.2 [**Information interfaces and presentation**]: User Interfaces - *Theory and methods, User-centered design*; I.3.6 **[Computer Graphics**]: Methodology and Techniques - *Interaction techniques.*

INTRODUCTION
With the evolution of technologies, computing capabilities and rendering techniques, the use of 3D Virtual Environments (3DVE) is becoming popular. 3DVE are no longer restricted to industrial uses and they are now available to the mass-market in various situations: for leisure in video games, to explore a city in Google Earth or in public displays [26], to design house furniture [17] or to explore cultural heritage sites in a museum [6]. However, in these mass market contexts, the user's attention must be focused on the content of the message and not distracted by any difficulties caused by the use of a complex or inappropriate interaction technique. This is especially true in a museum where the maximization of the knowledge transfer is the primary goal of an interactive 3D experience. Common devices, such as keyboard and mouse [21] or joystick [30] are therefore widely used in museums. To increase the immersion of the user, solutions combining multiple screens or cave-like devices [6] also exist. However, these solutions are cumbersome and expensive.

Meanwhile, the Human-Computer Interaction (HCI) research domain is rapidly evolving and growing in complexity with new advanced forms of interaction such as mobile [16], ambient computing [13], spatial interfaces [15] and tangible user interface [29]. A common feature to these advanced forms of interaction is the attempt to involve and combine the use of multiple objects and entities taken in the physical and digital environments: interactive solutions are smoothly integrated in the user's activity and have been proved to be easier to apprehend by newcomers [24]. Successful uses of such advanced interaction have been recently demonstrated in mass-market applications involving 3DVE for museums [11][12].

It thus appears that 3D interactive applications are more and more widespread, from professional context to public spaces and from expert users to very occasional users. In addition, advanced forms of interaction techniques offer new potentials such as being based on personal belongings (devices or artefacts), integrated in the physical environment, easy to apprehend. But developing 3D interactive applications on one hand and advanced interaction techniques on the other hand are two preoccupations that are mostly considered through separated approaches, leading to compartmentalized progresses. There is therefore a need for understanding and supporting the engineering of advanced interaction techniques for exploring and taking advantage of 3DVE.

In this position paper, we first provide an overview on a new analytical framework for helping and guiding the design of advanced interaction techniques for 3DVE. We motivate and illustrate the choice of its grounding elements and then discuss a number of existing modeling approaches potentially useful to complement or refine this framework. We finally discuss our expectations from the workshop, with regard to the proposed framework and more widely with regards to the engineering of advanced interaction technique.

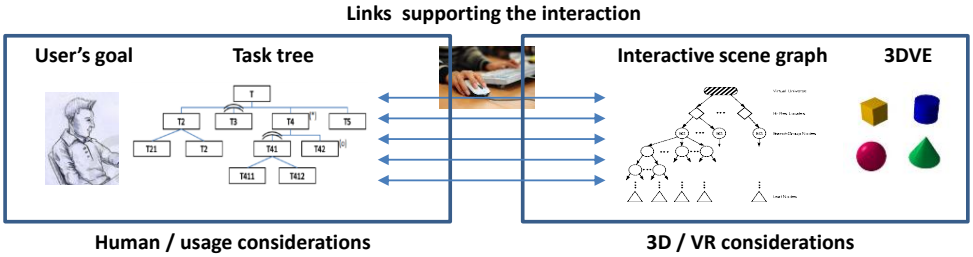


Figure 1: Overview of our analytical framework

OVERVIEW OF A NEW ANALYTICAL FRAMEWORK

We first present the two pillars of our modeling approach, tasks analysis and interactive scene graph. We then introduce the notion of links between these two pillars.

Tasks tree

The first pillar of the analytical framework is the result of a task analysis: a task tree. Task analysis (Figure 1 – left) consists in a decomposition of user's goal into tasks and sub-tasks which must be achieved to reach the user's goal. Task analysis is widely accepted in the HCI community as a starting point to the design of interactive techniques because it thus provides an algorithm of the user's activity, the logic and the dynamic of tasks accomplishment. But a task analysis does not express how the task is concretely performed with the system: no information related to the interaction technique is provided. Task analysis is particularly useful to understand and structure the user's activity, define functional specifications, identify data requirements, etc. [2].

Different formalisms exist to represent the result of a task analysis. We choose to rely on the familiar Hierarchical Task Analysis (HTA) formalism [2]. With HTA, the decomposition of the task is presented as a hierarchical tree of tasks and sub-tasks, enriched with some attributes (iteration, optional, parallel, etc.). Choosing a light approach ensures that non-experts are able to use it as a support for designing advanced multimodal interaction for 3DVE.

When used to describe tasks with an interactive 3DVE, most of the identified tasks are specific to the application domain and the leaves of the tasks tree are close to Bowman's tasks [5] (navigation, selection, manipulation, system control). But none of the 3D elements impacted by the tasks can be specified. This is where the interactive scene graph comes to action.

Interactive scene graph

The second pillar of the analytical framework is the interactive scene graph which provides a structured description of the 3DVE to be used. This scene graph is therefore specific to each 3DVE. A 3DVE generally consists of 3D objects (meshes, widgets or basic geometrical elements such as cone, cube, cylinder, etc.), lights and virtual cameras. At a finer grain 3D objects are described as a set of vertices (geometry), faces and edges (topology). Manipulations of 3D objects (translations, rotations or scaling) must therefore take into consideration the underlying topology and geometry. To assist this process, the concept of scene graph [25] has been developed to organize the 3D elements and provide for developers a structure for the assembly of a 3D scene.

Scene graph is a widely accepted method used in the Computer Graphic (CG) community to describe the essential components of a 3DVE. Scene graphs are also relevant to our context because we need to understand and take into account the structure of the 3D scene to design the interaction with it. But, we are not interested in elements related to the implementation of the scene graph by the 3D API in charge of the rendering. We are also not interested in the way the scene graph may impact the use of 3D engines for solving issues like texture management or collision.

However, with the scene graph description, only geometric and topologic aspects are expressed. It is not clearly identified which parts of which components of the 3D scene are likely to be impacted by user's interaction. To this end, we propose to define the "interactive scene graph" (Figure 1 – right). Its aim is to highlight and characterize **handled** and **not-handled** objects, i.e. objects impacted or not by one of the user's tasks identified during the tasks analysis.

The definition of the "interactive scene graph" is based on the most relevant features used to support 3DVE user's interaction in 3D engines like Unity 3D [32], Irrlicht [34]. It is also derived from the standard description language X3D [33]. X3D supports the description of ***animated*** 3D scenes: behaviors among 3D nodes are expressed in script nodes or simple links among 3D nodes. The role of the "interactive scene graph" goes beyond the description of animations in the 3DVE: it emphasizes which are the elements and attributes of the 3D scene with which external elements may ***interact***. Based on these existing approaches, we distinguish two types of handled objects: **components** and **renderers**.

Components are 3D objects composing the 3DVE (mesh, geometrical element, widgets). User's interaction may impact Components through the modification of two classes of attributes: ***state*** and ***manipulation***. State attributes refer to the color, the texture or the visibility (display or not in the 3DVE) of the object. Manipulation attributes are more complex. Three levels of manipulation attributes coexist:

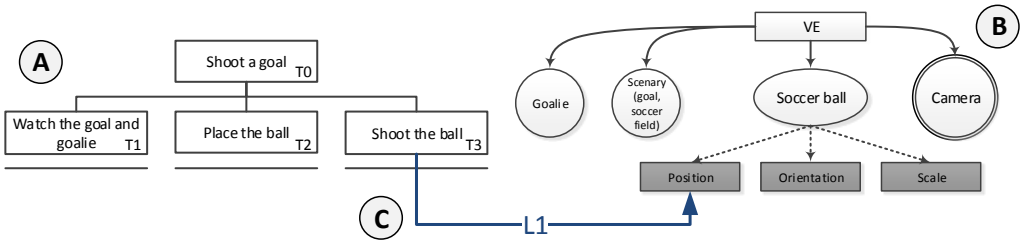


Figure 2: Task tree (A), scene graph (B) and link (C) of the Penalty Shootout interaction technique with Portico [4]

the *object*, its *faces* and its *points*. First, one may modify position, orientation or scale of an *object* as a whole. Second, an object is made of a set of *faces*: depending on the object structure, modifying a face can be limited in terms of degrees of freedom in orientations and scales. Third and finest level, a face of an object is made of a set of *points*: at that level one can only acts on the 3D position of each point. Faces and points levels are thus useful to refine and characterize the deformation of a 3D object.

Renderers are objects such as lights or camera, taking part in the rendering of the 3D scene. User's interaction may impact Renderers through the modification of two classes of attributes: ***state*** and ***manipulation***. States include attribute such as enabled/disabled and color. Manipulation attributes correspond to position and orientation, with orientation depicting the definition of the point of view of the camera.

The next step consists in identifying existing links between elements of the 3DVE and user's sub-tasks of the task tree that are affecting them.

Linking user's task with elements of the 3D scene

It consists in identifying for a set of sub-tasks of the task tree, the attribute(s) of the interactive scene graph affected by the realization of each of these specific sub-tasks (Figure 1 – middle). As a result pairs of user's sub-task and attributes of the 3DVE are clearly highlighted. This set of links provides a complete view on the user's interaction that will be performed in the 3DVE to perform the user's activities required to reach his/her goal. Each link depicts the use of one interaction modality [20]: on the one hand each link may involve a different interaction modality, on the other hand every links is using the same and unique interaction modality. In addition, we anticipate that the operators (sequence, alternative, etc.) present in the task tree at higher-levels will influence the design of the links.

Highlighting the pairs of user's sub-task and attributes of the interactive scene graph therefore constitutes the description of the overall user's interaction with the 3DVE. Furthermore, it establishes a link between user's activities (task analysis) and the 3DVE content and behavior (the interactive scene graph). Therefore, this description constitutes a support to reason about the design of the overall system and takes advantage of HCI and CG specificities. It overcomes the description of one specific task or one specific technique out of the context of use and manipulation of the 3DVE. In the next section, we illustrate the overview of our analytical framework with an example of the literature.

ILLUSTRATIVE EXAMPLE

Portico [4] is an interactive system for enabling tangible interaction on and around tablet computers. We focus our illustration on one of their application example named penalty shootout. Let us describe the interactive setting with the use of our analytical framework. The aim of the user is to shoot a goal (Figure 2 – A - task T0). For this, the user has to watch the goal and goalie (T1), place the ball (T2) and shoot the ball (T3). The system does not support multiple balls thereby there is a unique ball available for the user. Now, the interactive scene graph (Figure 2 – B) is composed by a **non-handled** camera (*renderer*), a **handled** soccer ball (*component*) and some **non-handled** objects like the goal, the goalie and the soccer field. Finally, the designer has adopted a tangible interaction to establish the link between the task tree and the scene graph (Figure 2 – C). Concretely, the soccer ball is a physical object manipulated by the user and thus, the sole task (L1) impacting the interactive scene graph is the task T3. The link L1 of this example connects the task T3 to the *manipulation* attribute position of the 3D ball in the scene graph. The task T2 does not impact the virtual ball position because the virtual ball position change only when the physical ball touch the tablet.

From this description, we can identify that the interactive situation supports the manipulation task in a 3DVE through the link L1 and an implicit selection task through the manipulation of the physical ball. The remaining Bowman's 3D tasks (navigation, system control) are not supported by this interactive technique. The description of the link L1 also highlights a direct connection between a physical object and a virtual 3D element (the ball): physical behavior and representation are directly mapped to the behavior of the corresponding digital object. Although it is a rather simple example, it shows that the framework can help in visualizing when the user's focus has to be on the 3D scene with regards to the user's task realization. If several links are present it may also help identify inappropriate sequence of interaction, such as switching modalities while focusing on the same 3D parts.

To better structure this kind of reasoning, designing the link L1 is subject to a set of design aspects that we extracted from the literature and summarize in the following section.

DESIGN ASPECTS OF THE INTERACTION WITH 3D

The HCI and CG communities have already been working on the design and implementation of advanced interaction techniques with 3D. Different points of view have been adopted thus revealing multiple design aspects. We summarize these considerations along the three parts of our analytical framework: user's interaction, 3D system and links between them.

Regarding the **user's interaction**, relevant considerations include the specification of the users expertise with the manipulation of the application as defined in Rasmussen work [22]. The definition of the application type (AR, VR, desktop application) is also considered as crucial in the 3-DIC model [10]. Given the use of advanced interaction technique, it is also required to specify which parts of the physical world are involved. It includes a description of the objects used and their constraints [27]. It also requires to specify how information are transferred from the user to the 3D system and vice-versa, as partly addressed in the ASUR [8] and MIM [7] models. It goes even to more precise description of the gestures and characterization of movements that will be performed by the users [19] as well as perceptual properties of interest such as visual, tactile and auditory properties.

Regarding the **3D system**, as already mentioned X3D [33] is the standard to describe the 3D nodes, structure and internal animation or behavior. Most of works however focus on the description of virtual reality interaction techniques such as ray casting and are focusing on their behavior or implementation. Among them the reusable library of 3D interaction technique [9], the Petri Net model [31], ontology model [14], the 3-DIC model [10] or IFFI [23] and Viargo [28] library are complementary alternatives.

Finally, in the literature, **the link between our two pillars** is often limited to the analysis of the required input device. Simple taxonomies offer an overview of the possibilities such as the Mackinlay taxonomy [18]. More elaborated models like RVDT [1] or InTml [9] deal with a particular aspect of input device, e.g. the type of data (float, integer, boolean) and the number of sensed DOF.

DISCUSSION

Obviously, developing advanced interaction techniques for 3DVE requires to confront multiple design considerations and to pay attention to both communities' preoccupations. To do so, offering a structured and refined set of design attributes that reconcile these multiple aspects will lead to a better understanding of the links between a user's goal and attribute of a 3D scene. For example metrics might be extracted to efficiently compare techniques; properties might be defined to clearly express how design choices in the user's part impact design choices in the 3D system parts and conversely.

We believe that providing such a structured approach to describe the links between task and scene graph is a fruitful way to help reason about the design and implementation of advanced interaction techniques for 3DVE. The resulting model or notation will constitute a pseudo-formal description language of interaction techniques for 3DVE. From such description, a semi-automatic implementation of the described advanced interaction for 3DVE could then be built in a platform for rapid development of multimodal interaction such as the Dynamo framework [3].

CONCLUSION

Applying advanced forms of interaction to 3D applications is required to contribute to a more effective use of 3D interactive environment. As multiple types of user and context are potentially targeted a user centered approach to the design of interactive 3D application is particularly expected.

In this paper we proposed a way to narrow two communities by involving well established design resource of each domain as the two pillars of a dedicated approach. We then identified a set of existing design and implementation supports for bridging the gap between these two pillars.

During the workshop, we hope to find the opportunity to further illustrate the use of this framework on different prototypes we have implemented in our lab. We then expect a fruitful discussion with the other participants of the workshop to identify additional existing design approaches relevant to this context of interaction with 3DVE, or relevant metrics, properties or considerations. In particular, we are interested to discuss what could be the ways to tightly anchor 3D specificities in the design of interaction technique. We are also interested in refining the links between our two pillars with relevant approaches. Finally, we hope to hear about similar approaches in different contexts, i.e. a context in which advanced HCI and another domain are involved and in which engineering supports of the two communities have been brought together. From such situation we expect to hear about lessons learnt, benefits and limits of such approaches.

REFERENCES

1. Ajaj, R., Jacquemin, C., and Vernier, F. RVDT: a design space for multiple input devices, multipleviews and multiple display surfaces combination. *In Proc. of ICMI-MLMI'09*, ACM (2009), 269–276.
2. Annett, J. Hierarchical Task Analysis. In D. Diaper and N. A. Stanton, eds., *The Handbook of Task Analysis for Human-Computer Interaction*. 2004, 667.
3. Avouac, P.-A., Lalanda, P., and Nigay, L. Autonomic management of multimodal interaction: DynaMo in action. *In Proc. of EICS '12*, ACM Press (2012), 35–44.
4. Avrahami, D., Wobbrock, J.O., and Izadi, S. Portico: tangible interaction on and around a tablet. *In Proc. of UIST '11*, ACM Press (2011), 347–356.

5. Bowman, D.A., Kruijff, E., LaViola, J.J., and Poupyrev, I. *3D User Interfaces: Theory and Practice*. 2004.
6. Christou, C., Angus, C., Loscos, C., Dettori, A., and Roussou, M. A versatile large-scale multimodal VR system for cultural heritage visualization. *In Proc. of VRST '06*, ACM Press (2006), 133–140.
7. Coutrix, C. and Nigay, L. Mixed Reality: A model of Mixed Interaction. *In Proc. of AVI '06*, ACM Press (2006), 43–50.
8. Dubois, E., Gray, P.D., and Nigay, L. ASUR++: A Design Notation for Mobile Mixed Systems. *In Proc. of MobileHCI'02*, Springer Verlag (2002), 123–139.
9. Figueroa, P. and Castro, D. A reusable library of 3D interaction techniques. *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, IEEE (2011), 3–10.
10. Figueroa, P., Dachselt, R., and Lindt, I. A Conceptual Model and Specification Language for Mixed Reality Interface Components. *VR 2006, In Proc. of the Workshop "Specification of Mixed Reality User Interfaces: Approaches, Languages, Standardization"*, 4–11.
11. Hornecker, E. Interactions around a contextually embedded system. *In Proc. of TEI '10*, ACM Press (2010), 169–176.
12. Huang, C.-R., Chen, C.-S., and Chung, P.-C. Tangible photorealistic virtual museum. *IEEE Computer Graphics and Applications 25*, 1 (2005), 15–17.
13. Ishii, H., Wisneski, C., Brave, S., et al. ambientROOM: integrating ambient media with architectural space. *CHI 98 conference summary on Human factors in computing systems - CHI '98*, ACM Press (1998), 173–174.
14. Latoschik, M.E. A user interface framework for multimodal VR interactions. *In Proc. of ICMI '05*, ACM Press (2005), 76–83.
15. LaViola, J.J. and Keefe, D.F. 3D spatial interaction. *ACM SIGGRAPH 2011 Courses on - SIGGRAPH '11*, ACM Press (2011), 1–75.
16. Lee, D., Hwang, J.-I., Kim, G.J., and Ahn, S.C. 3D interaction using mobile device on 3D environments with large screen. *In Proc. of MobileHCI '11*, ACM Press (2011), 575-580.
17. Lee, J.Y., Seo, D.W., and Rhee, G.W. Tangible authoring of 3D virtual scenes in dynamic augmented reality environment. *Computers in Industry 62*, 1 (2011), 107–119.
18. Mackinlay, J., Card, S., and Robertson, G. A Semantic Analysis of the Design Space of Input Devices. *Human-Computer Interaction 5*, 2 (1990), 145–190.
19. Manches, A., O'Malley, C., and Benford, S. Physical manipulation: evaluating the potential for tangible designs. *In Proc. of TEI '09*, ACM Press (2009), 77–84.
20. Nigay, L. and Coutaz, J. A design space for multimodal systems: concurrent processing and data fusion. ACM (1993), 172–178.
21. Pecchioli, L., Carrozzino, M., Mohamed, F., Bergamasco, M., and Kolbe, T.H. ISEE: Information access through the navigation of a 3D interactive environment. *Journal of Cultural Heritage 12*, 3 (2011), 287–294.
22. Rasmussen, J. Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering. (1986).
23. Ray, A. and Bowman, D.A. Towards a system for reusable 3D interaction techniques. *In Proc. of VRST '07*, ACM Press (2007), 187–190.
24. Shaer, O. and Hornecker, E. Tangible User Interfaces: Past, Present, and Future Directions. *Foundations and Trends® in Human–Computer Interaction 3*, 1-2 (2009), 1–137.
25. Strauss, P.S. and Carey, R. An object-oriented 3D graphics toolkit. *ACM SIGGRAPH'92 26*, 2 (1992), 341–349.
26. Tan, D.S., Gergle, D., Scupelli, P., and Pausch, R. Physically large displays improve performance on spatial tasks. *ACM Transactions on Computer-Human Interaction 13*, 1 (2006), 71–99.
27. Ullmer, B., Ishii, H., and Jacob, R.J.K. Token+constraint systems for tangible interaction with digital information. *ACM TOCHI 12*, 1 (2005), 81–118.
28. Valkov, D., Bolte, B., Bruder, G., and Steinicke, F. Viargo - A generic virtual reality interaction library. *2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, Ieee (2012), 23–28.
29. Williams, C., et al., P. TZee: exploiting the lighting properties of multi-touch tabletops for tangible 3d interactions. *In Proc. of CHI '11*, ACM Press (2011), 1363–1372.
30. Wischgoll, T. and Meyer, J. An explorational exhibit of a pig's heart. *ACM SIGGRAPH 2005 Posters on - SIGGRAPH '05*, ACM Press (2005), 138.
31. Ying, J. An approach to Petri net based formal modeling of user interactions from X3D content. *In Proc. of Web3D '06*, ACM Press (2006), 153–157.
32. Unity 3D - Game Engine. 2012. http://unity3d.com/. Accessed: April 2014
33. Web3D Consortium : X3D for Developers. 2012. http://www.web3d.org/x3d/. Accessed: April 2014
34. Irrlicht Engine - A free open source 3D engine. 2014. http://irrlicht.sourceforge.net/. Accessed: April 2014

Using Small, Simple, Wearable Devices for Large-Scale Data Gathering

Judy Bowen
The University of Waikato
New Zealand
jbowen@waikato.ac.nz

Annika Hinze
The University of Waikato
New Zealand
hinze@waikato.ac.nz

Sally-Jo Cunningham
The University of Waikato
New Zealand
sallyjo@waikato.ac.nz

ABSTRACT
Collecting real-world, long-term data from work environments is a challenging exercise. There are many different, and well-established, techniques from the fields of psychology and HCI (such as ethnography, auto-ethnography, diary studies, video studies *etc.*) that aim to address this. The choice of which to use is often determined not only by the nature of the data being collected, but also the availability and suitability of both personnel and equipment for the environment in which people are being studied. While the continuing emergence of hi-tech solutions (particularly in areas such as video and voice capture) can help with this by providing less obtrusive ways of capturing information, they can still be problematic for large-scale data-gathering experiments. We are currently working in the domain of the New Zealand forestry industry with the aim of building up a large repository of data of workers' activity levels throughout the day (both when at work and at home) and sleep patterns at night, with the aim of supporting safety efforts. In order to achieve this we have chosen to use a low-tech solution: using lightweight, wearable activity trackers with the aim of creating an extensive record of forestry workers' activities and well-being. Our goal is to use this data to identify potentially hazardous workplace situations.

INTRODUCTION
The recent developments in personal activity trackers have seen the simple pedometer redeveloped into a data-capture device capable of a much wider variety of capabilities than just counting steps. This has led to a proliferation of such devices, as well as the development of associated mobile phone and 'smart watch' applications that provide visualisations and analysis of activity data. While the target users for these devices are typically individuals interested in personal fitness or their own health and well-being, the fact that these trackers are low-cost (typically less than NZ$150 per unit) and lightweight makes them ideal research tools for large-scale data collection. A description of our research domain follows.

In 2013, fatalities in the NZ Forestry industry were twice the annual average NZ workforce rate of 5/year [9], and overall NZs highest rate of workplace injury deaths [3], with numbers increasing. The rate of ACC[1] claims for the forestry sector is almost six times the rate for all sectors and currently costs over NZ$2.3 million per year [4]. Both the severity of accidents and long-time injury frequency are increasing. The NZ forestry accident rate is more than 6 times higher than that of the UK and the death rate is 34 times higher.

Forestry is labour-intensive (most workers undertaking 40-60 hours of work per week). The tasks of tree felling and breaking out are the main activities contributing to serious accidents. There is a general consensus that pressure points include fatigue, dehydration, distraction, isolated work, remote locations, high staff turnover and production pressure, however no systematic data has ever been recorded to test these hypotheses. Major stakeholders (e.g., worker unions, government agencies, forestry management corporations) have conflicting views of the most significant contributing factors to accidents.

In a study conducted in 2010 video recordings of eight workers gave some insight into the differences between work patterns of novices and experienced tree fellers [10]. In the initial study the researcher, Parker, tried to film workers in their everyday work environment. However he soon found that what he was recording was not the usual work practices of the tree fellers, but rather the activities of tree fellers trying to prevent the researcher getting injured in their hazardous work environment, or getting in the way of their work activities.. Subsequently Parker developed a light-weight wearable camera which was used in the next study, but the cost and availability of the equipment meant that this study was limited to just eight participants. Because it has been so difficult to gather accurate data on worker activities, much of the current research into safety efforts for NZ forestry is focussed on developing robotic solutions (to remove workers from the equation) or on managing worker behaviours rather than investigating underlying causes.

In order to understand the causes (and therefore consider the prevention) of accidents we need a much better understanding of all of the contributing factors. Typically when an accident is reported only the immediate circumstances are considered relevant (what the worker was doing at the time the accident occurred) whereas we contend that there may be a series of

[1]New Zealand's accident compensation scheme, which provides financial compensation to New Zealand citizens, residents, and visitors who have been injured in an accident.

Figure 1. Polar Loop, FitBit Flex and FitBit One

contributing factors that have happened in the time leading up to the adverse event (and this timescale may be days before, rather than immediately prior to, the accident). To examine this hypothesis further, we need a way to capture data over extended periods of time from large groups of forestry workers with the data including all of their activities, not just what they are doing at work. So if (as is often stated) tiredness is a contributing factor, what is the cause of the tiredness? Is it solely the physical effort of the job (estimated as equivalent to running a marathon a day) or is this exacerbated by the fact that many of the workers are involved in sports outside of work and have perhaps been playing rugby for much of the weekend, or have poor sleep patterns, or have a two hour drive to get to the workplace each day etc.

RESEARCH OVERVIEW

The longer term aim of our research is to predict (and ultimately prevent) hazards by harnessing the power of a new generation of lightweight, wearable technology (so-called fitness trackers). We aim to create an extensive record of forestry workers' activities and well-being to build up a contextual history. Mining this record, we will identify hazard patterns and ultimately look at ways of encoding pattern detection algorithms within similarly lightweight, wearable devices to provide real-time hazard warnings.

Our work is currently structured into six components:

- Comparing different activity trackers and their suitability for collecting the type of data we are interested in
- Pilot studies in the field with different categories of workers and different devices
- Refining the nature/type/scope of data to be gathered
- Large scale data gathering
- Data analysis and development of safety models
- Development of technological solutions

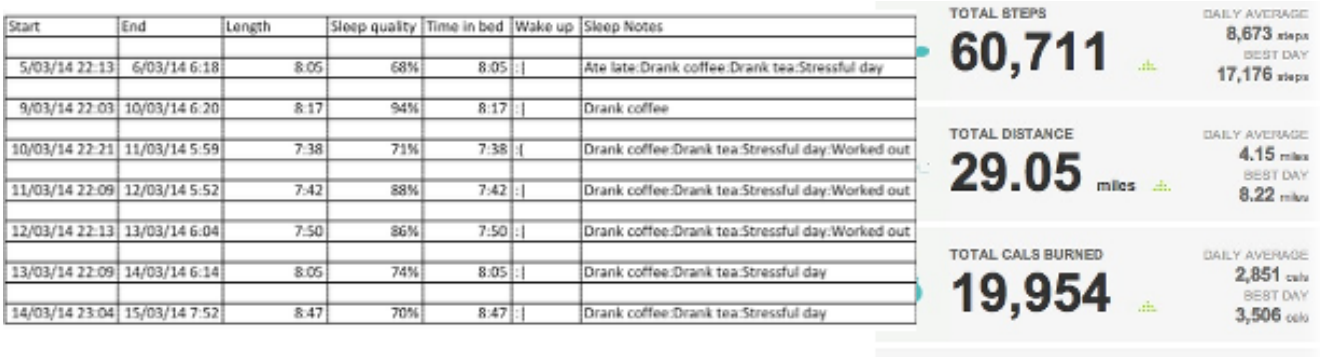
We are currently experimenting with activity trackers such as the FitbitFlex, Polar Loop, Fitbit One etc. shown in Figure 1. These have similar capabilities to each other including monitoring of steps taken by the wearer, identifying stair or hill-climbing activities, estimating calorie burn (based on steps taken and pre-set parameters for age and weight) and monitoring sleep patterns. The data produced is then provided to users by way of graphs and data logs (see Figure 2). The raw data can also be accessed via an API at a lower level of granularity, so it is possible to monitor activity on a minute by minute basis.

Our initial studies with these devices have enabled us to generate sets of data which we can use in our first experiments, and we are combining these with participant diary studies in an effort to match a user's record of their activity and well-being with the device data. We also have initial data regarding usability of the devices from the perspective of the user. This will enable us to consider things like how much effort is required to use the device (the less interaction our participants have with the device the better), how often the battery needs to be recharged, whether wearing the device has any effect on everyday activities, etc.

Our proposed solutions are based around our previous research expertise in the areas of ethnography [8, 7], formal modelling of interactive systems [1, 2], complex event processing [5, 6] and forestry safety [10, 11].

THE CHALLENGE

The major engineering challenge we are facing is how to use simple, cheap, lightweight, wearable devices (activity trackers) in smarter ways for large-scale data-collection. The availability of these devices and their low cost mean that they are an ideal choice for using in our research environment. We



Start	End	Length	Sleep quality	Time in bed	Wake up	Sleep Notes
5/03/14 22:13	6/03/14 6:18	8:05	68%	8:05	:|	Ate late:Drank coffee:Drank tea:Stressful day
9/03/14 22:03	10/03/14 6:20	8:17	94%	8:17	:|	Drank coffee
10/03/14 22:21	11/03/14 5:59	7:38	71%	7:38	:(	Drank coffee:Drank tea:Stressful day:Worked out
11/03/14 22:09	12/03/14 5:52	7:42	88%	7:42	:|	Drank coffee:Drank tea:Stressful day:Worked out
12/03/14 22:13	13/03/14 6:04	7:50	86%	7:50	:|	Drank coffee:Drank tea:Stressful day:Worked out
13/03/14 22:09	14/03/14 6:14	8:05	74%	8:05	:|	Drank coffee:Drank tea:Stressful day
14/03/14 23:04	15/03/14 7:52	8:47	70%	8:47	:|	Drank coffee:Drank tea:Stressful day

Figure 2. Standard Statistics for Sleep Cycle and FitBit Devices

can provide large groups of workers with the activity monitors and the devices are small and unobtrusive enough to be used over long periods of time to gather data. Conversely, many of these devices do not provide the ability to alter the parameters of the data being collected and cannot be programmed or extended to provide different kinds of feedback to users. We envisage that the simplest devices will be used for initial data gathering and experimentation, while more sophisticated devices (programmable smart-watches like the Pebble for instance) will be used for the final technological solutions.

Additional challenges of our work include:

- Identifying useful and informative patterns from large quantities of data
- Matching the patterns to hazardous situations
- Determining hazard predictions from the patterns
- Incorporating real-time alerting into small wearable devices

Of most interest here is the challenge of working with limited interactive devices rather than the more common problem of dealing with increasingly complex systems. In particular finding novel ways of making the most of their data capabilities as well as exploring options to enable them to act as warning mechanisms in a hazardous environment.

For our current domain of interest, the forestry sector, our goal is to use trackers which require minimal user interaction and which are as unobtrusive as possible, at the same time we want our final solutions to be able to collect a rich set of data. Adapting parameters to be captured and understanding how this can be achieved using these simple devices is an additional challenge we face. There are, of course, may other domains in which similar types of tracking may be useful and our work may be extended to investigate some of these as the project progresses.

REFERENCES

1. Bowen, J., and Reeves, S. Modelling user manuals of modal medical devices and learning from the experience. In *Proceedings of the Fourth ACM SIGCHI Symposium on Engineering interactive Computing Systems (Copenhagen, Denmark, June, 2012). EICS '12*, ACM, New York, NY (2012).
2. Bowen, J., and Reeves, S. Modelling safety properties of interactive medical systems. In *Proceedings of the Fifth ACM SIGCHI Symposium on Engineering interactive Computing Systems, EICS '13* (2013), 91–100.
3. Chief Coroner's office. http://www.justice.govt.nz/courts/coroners-court/publications/recommendations-recap/recommendation-recap-issue-4, 2014.
4. Department of Labour. Forestry sector action plan 20102013 (workplace health and safety strategy for New Zealand to 2015), August 2011.
5. Hinze, A., K. S., and Buchmann, A. Event processing: Applications and enabling technologies. In *International Conference on Distributed Event-Based Systems (DEBS)* (2009).
6. M. Kasi, A. Hinze, C. L., and Jones, S. Sepsen: semantic event processing at the sensor nodes for energy efficient wireless sensor networks. In *IACM Conference on Distributed Event-Based Systems* (2012).
7. Nichols, D. M., and Cunningham, S. J. Exploring social music behavior: an investigation of music selection at parties. In *In Proc 10th Int. Society for Music Information Retrieval Conference. ISMIR, Kobe, Japan* (2009), 747–752.
8. Nichols, D. M., and Cunningham, S. J. The use of paper in everyday student life. In *In Plimmer, B (ed/s) SIGCHI-NZ, Proc 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction (SIGCHI-NZ)* (2009), 65–68.
9. NZ Herald. New forestry fatality takes 2013 death toll to 10 : http://www.nzherald.co.nz/business/news/article.cfm?c_id=3 &objectid=11175056, 2014.
10. Parker, R. *Technological advances in the analysis of work in dangerous environments: Tree felling and rural fire fighting*. PhD thesis, Massey University, Palmerston North, 2010.
11. Parker, R. A stick insect that can fell trees. *New Zealand Logger magazine* (2013), p26–28.

Towards a Method, Techniques and Tools to Support the Development of Auditory Displays

Doon MacDonald and Tony Stockman
Interactional Sound and Music
C4DM
School of Electronic Engineering and computer Science
Queen Mary University of London
d.macdonald@qmul.ac.uk

ABSTRACT
Auditory displays have been around for some time, but lacking is any widespread understanding or consensus on how to develop and evaluate these interfaces. In the first part of this paper, we summarize the evidence for this lack of understanding and consensus, and go on to outline an approach to auditory display design based on Soundtrack composition. Space permits only an overview of the method, techniques and tools developed, but the presented method encapsulates the principles and ethos intended in applying ideas from soundtrack creation to auditory display design.

Author Keywords
Auditory Display, Auditory Interface, Soundtrack Composition.

ACM Classification Keywords
H.5.2. User Interfaces: Auditory Feedback, Theory and Methods

INTRODUCTION
Auditory display (AD) is the use of sound to communicate information from a computer to the user. The technique of using non-speech audio to convey information or perceptualize data is known as sonification. With a role similar to the use of visual displays in the human-computer interface, sound is employed to enable human-computer interaction, data exploration, accessibility and present aesthetics. ADs permit eyes-free usage by visually impaired users as well as by sighted users who need to use their sight for other tasks or who don't have line of sight of the display. The importance of sound in interaction design is growing as technology becomes increasingly embedded and portable (with smaller or even no screens) and the range of contexts of use continues to diversify. Some examples of auditory display included the Geiger Counter, which, as an early form of ubiquitous sonification [12], provided auditory clicks to display levels of radiation.

Audio has also been used for process monitoring, which often involves real-time and continuous audio playback in order to communicate directly to the user. For example, Cohen created the ShareMon system that alerted users to file sharing systems on an Apple Macintosh network [9]. Vickers created CAITLIN, a system that used audio feedback to help with program debugging [21]. Audio has also been used to represent the completion of tasks on the desk top, for example, Brewster designed a set of user interface widgets that used a collection of non-speech audio messages, known as Earcons, to enhance the visual feedback [6].

Some Challenges for Auditory Display Designers
- Sound is better at representing temporal information. The challenge, therefore, is representing spatial relationships with sound.
- Managing overlapping streams of information and avoid masking, whereby the perception of one sound is affected by the presence of another.
- Developing methods and tools to support the process of creating auditory displays.
- Taking an understanding of visual-HCI and seeing if it translates across to audio-HCI. Doing this by understanding what works best in the different mediums audio/visual.

MOTIVATION
The soundtrack of a film serves to support action; anchor meaning; develop thematic narrative strands; portray on and off screen action; enhance emotional reaction and generally give a sense of 'reality' to the real-world noises that objects make as they are moved, hit, and interacted with. The craft of soundtrack composition is well established and the creativity of the composer (sound designer) is supported and encouraged as a result of existing theoretical guidelines [7], techniques [20, 15], software-based tools, and, if the composer requires, optional sample and sound libraries. Soundtracks are composed and constructed with great intent and their success is apparent. It therefore makes sense to explore how this knowledge could be methodically applied to support the development of other applications that require sounds to be skilfully designed and arranged to display information.

ADs also serve to support action; anchor meaning; develop thematic content; represent both visual and non-visual unseen processes; enhance reaction and give a sense of reality to the real-world noises that virtual objects make as they are moved, hit, and interacted with. However, (unlike soundtrack composition) it is argued that many sonifications lack an aesthetic quality [22] and that the choices that went into the data/information - sound mapping lack rationale [14]. The results of the survey carried out by [14] revealed that many designers found it hard to incorporate audio due to a lack of existing structured methods, design tools and awareness of the possibilities. The authors concluded that as a result, design guidelines for auditory interfaces need to support a novice designer but also not limit the more experienced designer as well as supporting creativity and allowing designers to express and sketch out their initial ideas. It was also observed that design guidelines need to incorporate ideas and best practice concerning aesthetics.

Position

As a result of the above, we argue that the principles, tools and techniques of soundtrack composition can inform the design of a method for the creation of ADs. Specifically we propose that the application of such a method could support the iterative, creative choices that a designer will make from start to finish, as well as promote the creation of aesthetically considered ADs. As part of the process of developing the method, we aim additionally, to identify what sort of interfaces the method will be used to design. For example what will the function of the display be - monitoring? presentation of data for understanding and analysis? Feedback and acknowledgement of an interaction upon the interface, a combination?

Background

The idea of using music and soundtrack composition as well as other creative approaches from artistic practice to address issues concerning accessibility and aesthetics in the computer interface is not new. It was proposed by [10] that the use of sound within the arts could inform the use of sound for software development. The author argued that the role and function of sound in film as proposed by Film Critic and theorist Michel Chion [7] can take on a similar role and function within human-computer interaction: Namely the ability that sound has to alter perceptions of time and space as well as alter the perception of the speed and pace of movement. The author argued, by way of example, that the fact that sound can be designed to mimic texture, could enable an interface user to "feel" as though they have made contact with an interface object, even if it, in fact, remains unseen or virtual.

The perception of physicality that sound can give to an object or interaction was also described by [8]. The author argued that the overall feel and perception of an interaction can draw on the function of film sound to induce and communicate emotion. The author put forward the view that because music can create 'arousal' in the user, it can help the designer in focussing the user's attention, and as a result filter out noise-related distractions in the workspace. Additionally, the author pointed out that the use of repeating musical motifs can aid as an important cue to memory, which in turn could support the work flow by reducing dependence on visual communication.

The benefits of using sound to support work flow was highlighted by [3]. The authors argued that a conceptual and theoretical offering concerning how to use sounds for computerised instructional environments was needed. They argued that there are four ways in which the best practices of film sound can inform the sound design within computerised learning environments. These are as follows:

1. Consider sound from the start of the design process as opposed to after the visuals are put in place.
2. Have the means to 'listen' out for objects, actions, environments, emotions and physical or dramatic transitions within the narrative in order that they help shape the sound design.
3. Take into account the way that people hear and listen to the different types of sounds, albeit passively, actively or a combination [7, 22, 11].
4. Employ tools used on soundtrack composition [20] to support a methodical approach to incorporating sounds within the context of use.

OUR APPROACH

There have been several contributions toward establishing methods, techniques and tool kits for developing ADs [2, 18, 13]. With the aim to support novice AD designers, many of these are bought together and reviewed in [4, 5]. Interestingly, the authors conclude by recommending that a novice designer would benefit from "consulting someone with expertise in these methods". It was also argued by [14] that many of the methods are not accessible to an inexperienced designer and that, perhaps for this reason, there still is a general failure on the part of AD designers to take up methodologies and guidelines for the development of ADs.

The approaches reviewed above in the background section outline those contributions that have explored what arts practice can bring to the design of auditory interfaces. The research is rich in offering theoretical insights into the benefits of film sound and how an interface designer might be encouraged to explore its potential. However, despite previous research in this area, we argue that a designer contemplating how to apply principles of film sound to the creation of an auditory display, having researched any proposed approaches, would come away with inspiration, techniques and methods to support only part of their design, rather than the knowledge or confidence in how to create an entire display.

As a result we support the argument that creating a method for producing auditory display based on soundtrack composition can help an AD designer by offering a structured, accessible and supportive set of guidelines and techniques to assist them through the creative process. A distinguishing feature of our proposed method is that it is designed to encourage creativity, having its foundations in creative practice. In turn we speculate that the interfaces produced using the method may be somewhat characteristic, because they may manifest artistic

and creative qualities, making them more interesting and engaging to use [22].

The Method

To date we have developed the principles of the method and conceptualised the techniques that the method will support through a series of method steps and supporting tool kit. The steps are supported by use of a *Cue Sheet /Sound Map*[1], a *Database* and a *Time-line*. We have also carried out early stage evaluation of the work. We present these developments below.

Principles

We propose the following principles based on a synthesis of our investigations into soundtrack composition and the approaches that composers take [17], along side our investigations into designing auditory displays. We feel they encapsulate what the method should support and what they should enable the designer to achieve as a result of applying it.

1. Support the creativity of the designer
2. Draw upon the existing body of work and know-how surrounding soundtrack composition and approaches to designing ADs.
3. Support the design process from start to finish
4. Permit a designer to sketch out ideas and work in an iterative way
5. Utilise and make accessible the functions of a soundtrack and the proposed benefits of these to ADs and sonic interaction design
6. Support the production of a functional and aesthetically pleasing display

Method Steps

The following steps are designed to support the process from start to finish and combine techniques from human-computer interaction and soundtrack composition

- Method Step 1- Scenario Analysis: the designer is presented with the scenario and specific use-case for an initial read through. The use case outlines the user and the story of their interaction with the interface
- Method Step 2 - Interface and Interaction design: the designer analyses the scenario in terms of the users, the mode of listening they employ [11], the context of use and the initial interaction triggers.
- Method Step 3 - Information Design: the designer analyses the scenario in order to gather, categorise and map out information and data requirements within the interface. The results of this step are used to fill out the details of a Cue Sheet.
- Method Step 4 - Mapping information and interaction to sound: the outputs from the Cue Sheet/ Sound Map map directly to a database of sound design ideas and the designer can prototype various information to sound mappings.
- Method Step 5 - Iteratively auditioning and reorganising: the designer places the sounds on a time line in a first cut prototype of the sequence of interaction. The step then proceeds with the designer iterating through a process of auditioning and reorganising the sounds, taking into account the fact that certain interaction sequences may cause sounds to be rendered in parallel and/or in different sequences. The objective of this step is to examine the aesthetics of the sounds as they may be heard both individually in different interaction contexts and in relation to one another.

Techniques & Toolkit

Our method supports the designer in carrying out the method steps accounted above through a set of techniques that can be supported by the following: a *Cue Sheet /Sound Map*, a *Database* and a *Time-line*. To date we have begun an implementation of these in the graphical-programming language Max/MSP, as a set of tools that can be applied to support the designer.

The Cue Sheet/ Sound Map supports a narrative approach to both the gathering and the mapping of the requirements for the sound design for the auditory display (method steps 1-3). By utilising the cue sheet the designer is supported in the following ways:

1. Considering the story of the user interaction with the interface[19]; the context of use for the interface alongside the mode of listening that the user will employ in order to perceive and comprehend the sound.
2. Identifying characters, objects, locations, actions, themes, emotions and transition points in the story [20] in order that these can mapped to sound design ideas that support how these can be communicated most effectively through sound.
3. Identifying the physical properties of the information that needs to be sonified as well as the nature of how this might change as a result of changes in the interaction or data
4. Considering the story of the actual sound itself in terms of how it evolves in relation to the story it is telling [1]. The means to apply a linear, narrative-based who, what, where why, when structure to inform when the sounds occur, how long they occur for, how they change over time and what causes them to change.
5. Taking into account the non-linear nature of human-computer interaction by considering the transition points in the story. Designing sounds to smooth these points in order that the sound remains consistent [1]. In turn being able to design sound for several possible use-cases for one given scenario.

The Database acts as a knowledge base of sound design guidelines to support a specific function within a soundtrack.

[1]According to [20] A Cue Sheet (or sound map) acts as a guide for the sound team follow the different tracks of dialogue, music and sound effects. The tracks are represented along a time-frame of seconds and minutes and laid out horizontally. A "time-frame of dramatic sequences and with notations of sonic elements that may enhance the storytelling aspects of the film"

The resulting output of the Cue Sheet/ Sound Map directly maps to the different categories within the Database (method step 4). The categories are largely based on musical styles; structures; components and parameters.

The Time-line serves as an interactive tool to support the placement, audition and arrangement of the sounds within the display. The Time-line parallels and further supports the cue sheet in that it also caters for the non-linear nature of human-computer interaction, with the specific ability to focus on the affect this has on the sounds. The designer is encouraged to arrange and audition sounds according to the particular use-case under consideration (method step 5). The time line also encourages a potential investigation into the aesthetics of different sound sequences.

Initial Evaluation

We have carried out a preliminary evaluation of an early stage version of the Cue Sheet /Sound Map, which indicated its value as an organising mechanism in the AD design process, but which also highlighted the need for tight integration with a knowledge base of sound design principles, which guided our further development of the method and associated techniques and tools [16].

CONCLUSION

The creation of auditory displays for the interface is an ongoing challenge. Among the many future challenges for researchers are the design and evaluation of sound for mobile devices; the use of spatial audio and the consideration of increased and wide-spread usability and personalisation of interactions. In this paper we have presented our ideas towards a method, techniques and a tool kit to support an interface designer in applying audio. The method presents an integration of techniques employed in human-computer interaction design, such as user scenarios, with approaches and tools utilised in soundtrack composition. We hope that this poses an interesting design position and raises awareness of both the importance and the challenges involved in designing audio for the interface.

ACKNOWLEDGMENTS

This work is supported by the Media Arts Technology Programme, an RCUK Doctoral Training Centre in the Digital Economy.

REFERENCES

1. Back, M., and Des, D. Micro-narratives in sound design: Context, character, and caricature in waveform manipulation. In *Proc. of the Int. Conf on Auditory Display (ICAD)* (1996).
2. Barrass, S. TaDa! demonstrations of auditory information design. In *Proc. of the 3rd Int.Conf. on Auditory Display (ICAD)* (1996).
3. Bishop, M. J. Designing with Sound to Enhance Learning: Four Recommendations from the Film Industry. *The Journal of Applied Interactional Design 2*, 1 (2013), 5–16.
4. Brazil, E. A review of methods and frameworks for sonic interaction design: Exploring existing approaches. In *Proc. of CMMR/ICAD* (2009), 41–67.
5. Brazil, E., and Fernström, M. Subjective Experience Methods for Early Conceptual Design of Auditory Displays. In *Proc. of the Int. Conf. on Auditory Display (ICAD)* (2009), 1–8.
6. Brewster, S. The design of sonically-enhanced widgets. *Interacting with Computers 11*, 2 (1998), 211–235.
7. Chion, M. *Audio Vision. Sound on Screen*. Columbia University Press, 1994.
8. Cohen, A. The Functions of Music in Multimedia: A cognitive Approach. In *Proc. of the Int. Conf on Music Perception and Cognition* (1998).
9. Cohen, J. Monitoring background activities. In *Auditory Display*, G. Kramer, Ed., vol. XVIII. Santa Fe Institute, Studies in The Sciences of Complexity Proceedings, 1994, 499–532.
10. Cooley, M. Sound+ image in computer-based design: learning from sound in the arts. In *Proc. of the Int. Conf. on Auditory Display* (1998), 1–10.
11. Droumeva, M., and McGregor, I. Everyday Listening to Auditory Displays: Lessons from Acoustic Ecology. In *Proc. of the Int. Conf on Auditory Display (ICAD)* (2012), 52–59.
12. Ferguson, S. Sonifying everyday : Activating everyday interactions for ambient sonification systems. In *Proc. of the Int. Conf. on Auditory Display* (2013).
13. Frauenberger, C. *Auditory Display Design. An Investigation of a Design Pattern Approach*. PhD thesis, Queen Mary University of London, 2009.
14. Frauenberger, C., Stockman, T., and Bourguet, M.-L. A survey on common practice in designing audio in the user interface. In *Procs. of HCI 2007* (2007).
15. Holman, T. *Sound for Film and Television*, third ed. Elsevier/Focal Press, 2010.
16. MacDonald, D., and Stockman, T. The development of a method for designing auditory displays based on soundtrack composition. In *Proc. of the Int Conf. on Auditory Display* (2013).
17. MacDonald, D., and Stockman, T. Toward a method and toolkit for the design of auditory displays, based on soundtrack composition. In *Proc. of Human Factors in Computing (CHI)* (2013).
18. Mitsopoulos, E., and Edwards, A. A principled design methodology for auditory interaction. *Human-Computer Interaction - INTERACT '99* (1999).
19. Pirhonen, A., Murphy, E., McAllister, G., and Yu, W. Non-speech sounds as elements of a use scenario: A semiotic perspective. In *Proc. of the Int Conf. on Auditory Display* (2006).

20. Sonnenschein, D. *Sound Design: The Expressive Power of Music, Voice and Sound Effects in Cinema*. Michael Wiese Productions, 2002.

21. Vickers, P. *CAITLIN: Implementation of a Musical Program Auralisation System to Study the Effects on Debugging Tasks as Performed by Novice Pascal Programmers.* PhD thesis, Loughborough University, Loughborough, Leicestershire, September 1999.

22. Vickers, P. Sonification for Process Monitoring. In *The Sonification Handbook*, T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Publishing House, Berlin, Germany., 2011, ch. 18, 455–491.

Developing Methods and Techniques for the Design of Cross-modal Interfaces

Metatla O., Martin F.
School of Electronic Eng. & Computer Sci.
Queen Mary University of London
{o.metatla, f.martin}@qmul.ac.uk

Stockman T., Bryan-Kinns N.
School of Electronic Eng. & Computer Sci.
Queen Mary University of London
{t.stockman, n.bryan-kinns}@qmul.ac.uk

ABSTRACT
There exists relatively little work on the design of cross-modal interfaces, that is, interfaces which support collaboration between individuals that use different sets of modalities to interact with each other. In this paper, we examine the role of design patterns based on two phases of a design process. Firstly, we examine the role of participatory design workshops in identifying patterns that arise out of conflicting requirements between different interaction modes. We then describe how an analysis of these conflicts can lead to pattern-based solutions to interactional and implementation issues in the design of cross-modal displays.

Author Keywords
Guides; instructions; author's kit; conference publications; keywords should be separated by a semi-colon.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Cross-modal interaction is fundamental to human perception, involving the coordination of information received through multiple senses to establish meaning [10]. An example of this is when we both see and hear someone talking and associate the words spoken with the speaker, thus combining information received from two signals through different senses. In the design of interactive systems, the term cross-modal interaction has also been used to refer to situations where individuals interact with each other while accessing a shared space through different modalities such as graphical displays and audio output [11, 9]. In this paper, we examine how design patterns for cross-modal collaboration can be identified. We describe how we used an approach based on activity patterns [6] to uncover design patterns from two phases of a typical design process. First, we examine the role of participatory design workshops in identifying patterns that arise out of conflicting requirements between different interaction modes. We then demonstrate the application of activity patterns by reflecting on the evaluation phase of a cross-modal tool that supports collaborative diagram creation and editing by visually impaired and sighted coworkers. We also show how an analysis of the conflicts revealed by an activity theory-based analysis of these patterns can lead to solutions to interactional and implementation issues in the design of cross-modal displays.

BACKGROUND
Despite significant progress in the use of the audio and haptic modalities in interaction design, research into cross-modal interaction has so far remained sparse. Initial investigations have nonetheless identified a number of issues that impact the efficiency of collaboration in cross-modal settings. For example, an examination of collaboration between sighted and visually impaired individuals on an interactive puzzle game highlighted the importance of providing visually impaired collaborators with a continuous display of the status of the shared game [12]. Providing collaborators with independent views of the shared space, rather than shared cursor control, was also found to improve orientation, engagement and coordination in shared tasks. In another study, a multimodal system combining haptic devices with speech and non-speech auditory output was used to examine collaboration between pairs of visually impaired users on graph reading tasks [8]. Results showed that the use of haptic mechanisms for monitoring activities and shared audio output improves communication and promotes collaboration.

Although scarce, the literature on cross-modal collaboration has begun to generate insights into the knowledge that is needed to come up with effective designs to support interactions involving individuals with differing perceptual abilities across various domains. We propose to use design patterns as a means to capture such knowledge so that it can be effectively leveraged to provide solutions to support accessible collaborative working.

APPROACH

Theoretical foundation
We consider activity patterns [6] as a potential guiding framework for identifying and implementing design patterns for cross-modal displays. According to this framework, Alexander's patterns [1] could be appropriated to embody the principals of Activity Theory (AT) and hence could be used to analyse activity in terms of understanding tool-mediated work in its context [2]. AT views human activity in terms of a system of tool-mediated actions carried out by a subject (i.e. an individual or a group of individuals) in order to achieve a desired outcome. Actions are characterised in terms of how they are organised within a community context, and how they are regulated by internal rules and mediated by a division of labour. This unit of analysis is conventionally represented by a triangular model to show how its elements interact with each other (see Figure 1).

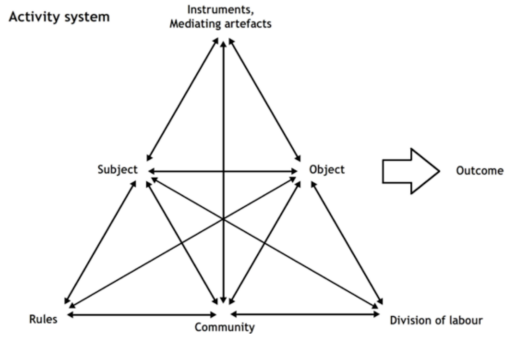


Figure 1. An activity system as conceptualized by activity theory.

According to [6], there are parallels between the design patterns principles as introduced in architectural design [1] and those of human activity as conceptualised by AT's unit of analysis. These include; the definition of a pattern in terms of three related components expressing the relationship between a given context, a problem and its solution, which is consistent with the method of AT; and the characterisation of a problem in a given context as being caused by a system of conflicting forces that arise in that context, which could be captured through AT's conceptual tool of contradictions. Additionally, the hierarchical levels of activity in AT can also be used in a similar way to the Alexandrian concept of scales to help structure the scope covered by activity patterns. The concept of activity levels in AT are hierarchically structured into three levels; activities, actions and operations where a given activity is realised through a set of concrete actions, which are in turn accomplished through a series of operations. This gives patterns a sense of scale from high-level activities down to low-level operations. According to [6], patterns could be written to reflect each element in a given activity system - the design of mediating artefacts; the work of a subject; the rules and procedures; and the roles within the division of labour or community of the work group - as well as organised into a coherent pattern language that preserves the unity of these elements within each of the three levels of activity.

In order to assess the feasibility of this organisational framework in supporting the process of identifying design patterns in cross-modal design, we applied it to data gathered from two phases of a typical process; requirements capture through participatory design workshops, and evaluation.

Initial participatory design workshop

The first stage of our approach involved setting up an initial workshop with 8 to 10 participants drawing from a network of users in the particular domain of focus. The workshop was organised around three main activities; focus group discussions, technology demonstrations, and audio-haptic mock-ups design. The aim of the focus discussions is to identify current best practice in the domain of concern, how current access technology supports this and a list of tasks which are either difficult or not possible using current access solutions. The technology demonstrations involve presenting a range of candidate technologies that could be used as a basis for designing solutions to the issues identified in the focus discussions. Visually impaired users often have a good knowledge of the access solutions they personally use, but do not necessarily have direct access to or experience of other relevant technologies. It is important that the capabilities of a given technology are demonstrated without reference to an actual application. For example, in order to ensure an application-independent demonstration of the haptic devices, we used a custom program that allowed us to switch between different effects that could be simulated with these devices, such as vibration, spring effects and viscosity. The custom program allowed us to manipulate various parameters to demonstrate the range of representations and resolutions that could be achieved with each device in real-time. In the audio-haptic mock ups design phase of the workshop, we asked participants to think through new designs, having had hands-on experiences of the capabilities of the candidate technologies. In this phase, participants worked in small groups including one or two members of the design team to identify technology solutions to the problems that arose in the focus discussions. To close the session, participants presented the audio-haptic mock-ups they constructed with their group to the rest of the participants for further discussion.

An unexpected outcome of these initial workshops was that the invited users spontaneously agreed to sign up to a email list which from their point of view provided a forum for the sharing of best practice and workarounds, and for us provided a community forum for the discussion of design issues and a user group we could draw upon for participants in later formative evaluations.

We used the activity patterns approach to drive retrospective analysis of data gathered from the workshop. Participatory design activities generate a huge amount of data and patterns could help with the process of organising the themes that emerge from this data. The concept of contradictions can be a useful guide to identify the tensions that exist in activity systems constructed to model the requirements and scenarios described by workshop participants, and hence could lead to insights about design solutions that could resolve such contradictions.

Example: Conflicting requirements

In an example of such scenarios, while discussing his experience of working with sighted colleagues and clients, a visually impaired producer described his frustration with the inaccessibility of graphical and diagrammatic representations used in digital audio workstations. The visually impaired producer explained how his work with sighted colleagues often involves exchanging projects back and forth in order to complete sub tasks involving the manipulation of audio captured using inaccessible formats or requiring interaction with inaccessible audio plugins. In some cases, these accessibility issues have led to his exclusion from potential collaborative projects because the standard formats used are not readily accessible or would take too long to work with.

In a second scenario, a visually impaired participant who specialises as an accessibility trainer described a similar experience with inaccessible visual tools. In this case, the issue was specific to working on collaborative projects that were coded

using an audio programming language known as Max/MSP, which is a visual programming language that uses diagrammatic representations as its main coding components. The visually impaired participant highlighted how inaccessible such programming languages are even though they are used to code audio, which could be considered a natural working modality for visually impaired individuals.

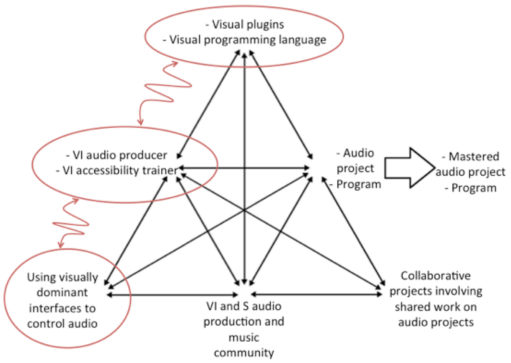


Figure 2. An activity system showing some contradictions captured through the PD workshop. Contradictions are highlighted with red circles and wavy arrows.

Both scenarios above could be captured by the activity system shown in Figure 2. Here, there is a clear contradiction between the subjects of the activity, i.e. the visually impaired audio producer and accessibility trainer, and the tools available to them as mediating artefacts in the context of their activities. Capturing these contradictions allows us to think about possible design solutions to eliminate them, which could eventually lead to the development of fully articulated design patterns that embody such solutions.

Identifying design patterns through evaluation

We used the activity patterns approach to reflect on the design of a diagramming cross-modal tool, which we evaluated with visually impaired and sighted users. The tool combines a visual diagram editor with auditory and haptic capabilities to allow simultaneous visual and non-visual interaction. That is, two coworkers collaborate on shared diagrams by accessing and editing them through the visual modality (for the sighted user) and the combination of audio and haptic modalities (for the visually impaired user) [9].

We deployed this collaborative tool in various workplaces including a local government office and a charity organisation where visually impaired and sighted coworkers access and edit diagrams as part of their daily jobs [9]. In the following, we describe an example of applying the activity pattern approach retrospectively to analyse data that we gathered from one of the field studies and how this helped the process of identifying potential patterns for cross-modal design.

Example: Consistency of interaction steps

In this example, a visually impaired manager (VI) and their sighted assistant (S) at a local government office edit an organisation chart to reflect recent changes in managerial structures. At one point during the interaction, the pair decides to create a connection between two nodes on the chart diagram to highlight a relationship between an existing and a new position. They do this while discussing how the tool should be used to create this relationship.

To create a connection between two nodes using the non-visual audio-haptic editor, the visually impaired user must 1) browse the chart to locate the first node and select it, 2) browse the chart again to locate the second node and select it, 3) select the type of connection they wish to use and 4) issue a command to create the desired connection. To do the same in the graphical editor, the sighted user must 1) select the type of connection they wish to use from the graphical tool bar, 2) select the first node on the chart, 3) drag the connection towards the second node using the computer mouse, and then 4) release the mouse to create the connection. Following the activity patterns approach, the actions of creating a connection between two nodes using the visual and non-visual editors can be represented as the two independent activity systems shown in Figure 3, which highlights a contradiction between the operational rules in the two activity systems; there is a mismatch between the interaction steps that each collaborator has to follow in order to create a connection between two nodes on the chart. Modelling the collaborative action of creating a connection in this manner has therefore uncovered a potential design flaw - which manifests itself as contradictions - that could hinder collaboration. Addressing this design flaw could lead to a design pattern that can eliminate the issues raised by the contradictions. For instance, the potential design pattern could describe the need to reconcile the two mediating artefacts in this context by ensuring *Equivalence and Consistency of Interaction Steps* between the visual and non-visual modalities.

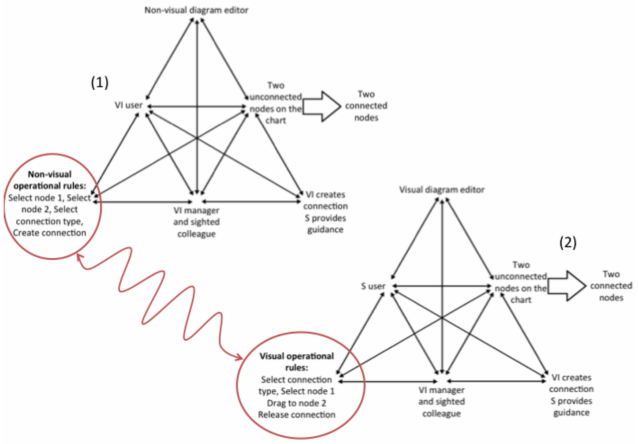


Figure 3. Two activity systems for creating a connection between two nodes using the non-visual (1) and the visual (2) editors. Contradictions are highlighted with red circles and wavy arrows.

Interaction and implementation design patterns

The conflicts highlighted when applying the activity patterns approach have also led us to identify a number of design patterns that could be used to guide the development of cross-modal software systems. Often, we found that interactional needs must be echoed in the actual implementation of such

cross-modal systems. In particular, we explored how existing software design patterns (e.g. [4]) could be extended in order to accommodate needs that are specific to cross-modal interface implementation. Here too, the conceptual tool of contradiction has helped our analysis and guided the process of matching interaction and implementation design patterns.

Example 1: Managing input and output conflicts

For instance, cross-modal software applications must be able to manage multiple input and output streams through various controllers. The Observer pattern is particularly suitable for this kind of applications [4]. That is, input and output streams can be easily modelled as controllers [1] and observers respectively. However, this creates potential conflicts when users attempt to interact with shared content. Care must therefore be taken in order to avoid situations where changes in one modality trigger unwanted effects in another.

An example scenario where the above issue could arise is when editing the positions of objects on a interface. In our cross-modal application, moving diagram items can be accomplished using a mouse drag-and-drop in the visual modality. In audio, the same task could be accomplished by selecting an item and using the keyboard arrow keys to change its position. Visual feedback of movement is realised by refreshing the visual display and updating the coordinates of the item in real-time, whereas that of audio is realised by displaying audio feedback in response to each keyboard input stroke.

The refresh rate of mouse input is typically sampled at a high rate in order to ensure smoothness of movement, which in turn yields a high volume of data coordinates. But while graphical displays can cope well with such high frequency updates, an auditory display may end up with a fairly high quantity of sounds to display which, if not managed, could amount to pure noise. We have employed an analysis of contradictory conflicts using the activity patterns approach to derive a variation on the Observer pattern that we dubbed *Cross-Modal Observer*. This pattern is an example of how an interactional need - in this case having equal access to a shared space - requires carefully engineered implementation. The Cross-Modal Observer pattern builds on the original Observer pattern by introducing a reference to the source of the input controller that could then be used to filter out unnecessary data output streams. Thus, unnecessary output, such as displaying audio in response to actions issued through the graphical display, can be filtered out to enhance the usability of the auditory display as well as the overall performance of the cross-modal system.

Example 2: Supporting awareness

Another example of where an interactional need could be echoed in, and hence help establish a cross-modal implementation pattern is the need to support awareness. Maintaining awareness is critical in group collaboration [5] and should therefore be adequately accounted for in cross-modal collaborative systems.

[1]This term comes from the *Model-View-Controller [7]* software design pattern.

In a collaborative system, accessing shared resources concurrently can lead to inconsistencies in the underlying data model representation. In order to address this, a locking system can be used to synchronise the interaction by forcing any user-issued editing command to acquire a lock on a piece of data before changing it. In our collaborative application, this meant that every edit message must be preceded by another message carrying a lock request. Once finished with the data, a lock release message is then sent to the server in order to free resources.

When requesting/releasing a lock, the minimum information that a message must contain is that of the source as well as the specific item that needs to be locked/unlocked for editing. From the point of view of the interaction, this information specifies the subject and the object of the editing action. Since the lock request and release messages are sent both when an edit action is initiated and terminated, the locking system mirrors the interaction of the user with the system, and can thus be used to convey awareness information about each user's actions to other collaborators. We thus derived a design pattern which we dubbed *Lock Driven Awareness* to implement and support awareness in cross-modal collaborative displays. This pattern allows developers of cross-modal collaborative systems to leverage the locking technique, which is often needed to manage access to shared resources. Additionally, information about the source of an action and its object of interaction could be augmented to include more detailed awareness information, such as the type of action and how it affects the content: move, rename, delete etc.

DISCUSSION

The design of cross-modal collaborative systems presents a unique set of challenges because such systems must allow individual users to equally contribute to the shared tasks while accommodating their individual perceptual differences. To date, no research has examined how to capture the knowledge required to design technology that makes cross-modal collaboration easier.

In our approach, the participatory workshops played a key role in identifying barriers to collaboration in the respective domains and identifying potential solution stems for these problems. The concepts of conflicting forces and contradictions in AT have proved valuable in highlighting mismatches or incompatibilities in cross-modal interaction. Typical mismatches we have encountered include:

1. Mismatches in the series of actions required to achieve the same result through different interfaces to a system.
2. Mismatches between the representation of actions performed in one interface and the way in which those actions are represented in another interface.

Having used AT-based analysis to uncover the mismatches or contradictions, we have found that further analysing the contradiction with the aim of defining a pattern solution can lead to patterns at either the design or implementation level which can remove the issues raised by the contradictions. Furthermore, pattern solutions derived in this way can be sufficiently abstract to be considered as providing useful guidelines for

design beyond the specific domain of interest. For example, ensuring *Equivalence and Consistency of Interaction Steps* between different interaction modes, and implementing *Lock Driven Awareness* to implement and support mutual awareness in collaborative systems, are design techniques which appear appropriate and applicable beyond the design domain of diagramming systems. Design patterns and pattern languages have been shown to facilitate the capture, presentation and communication of design knowledge [3]. We postulate that designers and application domain experts in cross-modal displays could benefit from using an activity patterns approach as both a means to identify and address design issues as well as a uniform representation for the design knowledge they generate.

CONCLUSION

We proposed that designers and application domain experts in cross-modal collaboration could benefit from using design patterns as a uniform representation for expert knowledge. To this end, we explored the question of how potential patterns can be uncovered from an iterative design process and suggested that activity patterns could be used as a structured method to address this question. One of the key benefits of using activity theory to identify patterns is the conceptual tool of contradictions, which can be a useful guide for designers to identify the tensions that exist in their designs when used in context and modelled as activity systems. We have exemplified how this approach was useful for us in managing requirement data from participatory design workshops as well as in the evaluation phase of a cross-modal collaborative tool. We plan to use this approach to both generate and articulate an initial set of patterns to form a pattern language for designing cross-modal collaboration, which we will then validate by applying the patterns in future design iterations and incorporating them in future studies and design and evaluation activities.

REFERENCES

1. Alexander, C., Ishikawa, S., and Silverstein, M. A pattern language: Towns, buildings, construction (center for environmental structure series).
2. Bertelsen, O. W., and Bødker, S. Activity theory. *HCI models, theories, and frameworks: Toward a multidisciplinary science* (2003), 291–324.
3. Borchers, J. O. A pattern approach to interaction design. *AI & Society 15*, 4 (2001), 359–376.
4. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
5. Gutwin, C., and Greenberg, S. Design for individuals, design for groups: tradeoffs between power and workspace awareness. In *Proc. of the 1998 ACM conference on Computer supported cooperative work*, ACM (1998), 207–216.
6. Guy, E. S. "...real, concrete facts about what works...": Integrating evaluation and design through patterns. In *Proc. of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '05 (2005), 99–108.
7. Krasner, G. E., Pope, S. T., et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming 1*, 3 (1988), 26–49.
8. McGookin, D., and Brewster, S. An initial investigation into non-visual computer supported collaboration. In *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, ACM (2007), 2573–2578.
9. Metatla, O., Bryan-Kinns, N., Stockman, T., and Martin, F. Supporting cross-modal collaboration in the workplace. In *Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers*, British Computer Society (2012), 109–118.
10. Spence, C., and Driver, J. Cross-modal links in attention between audition, vision, and touch: Implications for interface design. *International Journal of Cognitive Ergonomics* (1997).
11. Winberg, F. Supporting cross-modal collaboration: Adding a social dimension to accessibility. In *Haptic and Audio Interaction Design*. Springer, 2006, 102–110.
12. Winberg, F., and Bowers, J. Assembling the senses: towards the design of cooperative interfaces for visually impaired users. In *Proc. of the 2004 ACM conference on Computer supported cooperative work*, ACM (2004), 332–341.

High assurance interactive computing systems

José Creissac Campos
Departamento de Informática, Universidade do Minho & HASLab / INESC TEC
Braga, Portugal
jose.campos@di.uminho.pt

ABSTRACT
If interactive computing systems development is to be considered an engineering discipline, we need methods and tools to help us reason about and predict the quality of systems, from early in the design process. This paper provides a brief overview of work we have been carrying out in the general area of evaluating and ensuring the quality of interactive computing systems. Some of the work currently being carried out is also discussed. Discussed approaches range from the formal verification of user interface models through model checking, to the reverse engineering and model based testing of implemented interactive computing systems.

INTRODUCTION
Software has become present in all aspects of our lives, from safety critical applications, such as medical devices and the cars we drive, to social networks and games running on our phones. As the losses and disruption caused by software failures rise in severity, so does the need to guarantee that software will function *correctly*. This is a particular challenge for interactive computing systems, given not only the presence of the human factor, which must be considered during the analysis, but also the continuous evolution of interaction and implementation technologies, which make it difficult to asses, *a priori*, the quality of a system at development stages. Nevertheless, if the development of user interfaces is to be an engineering discipline, it must have techniques and tools to enable this analysis during development so that quality can be measured and predicted.

This paper briefly describes work that we have been carrying out to support the analysis of interactive computing systems. Building on that work it then outlines our views on some future lines of research. The rational behind the proposals for future work is that we need techniques and tools that better fit the typical development process of interactive computing systems. That is to say, techniques and tools that enable us to analyse the systems as they are developed. To achieve this we need means of leveraging, not only the analysis of any models that might have been produced during development, but also of the source code produced.

PREVIOUS WORK
One major goal of our work is to support the exhaustive, systematic and, and much as possible, automated analysis of interactive commuting systems. With this aim as the backdrop, a number of research direction have been pursued and are now discussed. A common trait of all the approaches is the focus on the system. That is, the main requirement is that either a model of the system, or the actual system, is available for analysis. An alternative approach, not explored here, would be to focus on cognitive models of the user (see, for example, ACT-R [1] or PUM [19]).

In order to support the systematic and exhaustive analysis of user interfaces, we have developed the IVY workbench[1] [6]. The tool supports the formal verification of interactive computing systems using model checking [8]. It aims to cater for the full cycle of analysis, from modelling to interpretation of the verification results. Models are expressed in a domain specific language (MAL interactors), and properties for verification in Computational Tree Logic (CTL) [8]. IVY has been applied to a number of different systems, mostly in safety critical domains (for example, medical [5] and aerospace [18]). How best to fold considerations about the user into the analysis has been a recurring concern, and is typically done by making explicit assumptions about how the user will react to the user interface [4]. The goal being to guarantee that only cognitively plausible behaviors are considered during the verification.

IVY has proven suitable to analyze control panel and WIMP style interfaces, but less so when considering larger interaction contexts, such as when considering ubiquitous computing systems. With the APEX framework[2] [16] we have specifically targeted ubiquitous computing environments. The framework combines a modelling tool (CPN Tools [13]) with a 3D application server (OpenSimulator[3]), in order to combine formal verification and prototyping. On the one hand, the models capture the behavior of active objects in the environment (for example, sensors or public displays), and are amenable to formal verification of their behavior [15]. On the other hand, the use of the 3D application server for prototyping purposes enables an empirical assessment of the user experience of the systems. The approach supports a multi-level evolutionary prototyping approach, where simulate devices can gradually be replaced by their physical counterparts. The simulation itself can resort to models of device

[1] `http://ivy.di.uminho.pt` (last visited 09/04/2014)
[2] `http://ivy.di.uminho.pt/apex` (last visited 09/04/2014)
[3] `http://opensimulator.org` (last visited 09/04/2014)

behavior or be migrated to code in the target virtual reality environment.

Both of the above approaches depend on the development of models (although the multilevel nature of APEX also supports programming the virtual devices directly in the environment). This begs two questions. The first regards the availability of models for analysis, the second the extent to each the models faithfully capture the relevant features of the system once developed, given the specific analysis being carried out. This last issue impacts the validity of the analysis, and what it means of the actual system once deployed.

Regarding the first issue, while Model Based User Interface Development approaches (consider, for example, [3]) advocate the use of models, there are concerns about the quality (from a user's perspective) of the user interfaces developed in this fashion (mainly when automatic refinement of models into code is considered). Indeed, human-centered design approaches typically advocate a process based on iterative prototyping and testing [10]. Additionally, agile approaches focus much earlier on the production of code [11]. This means that it is not guaranteed the models will necessarily be available for analysis.

When models are available, and especially when the generation of the system is not automated, the question arises of how faithful a representation of the code the model is. This not only is a problem due to the need to guarantee that the code is correctly generated from the original model, but also to the need to guarantee that both ends of the development process (models and code) are kept synchronized. Indeed, a known problem of model based software engineering is maintaining the consistency between model and source code.

Considering the above, we have been exploring the use of model based testing to directly analyze actual running application [17]. Model based testing works by comparing the prescribed behavior of a system (captured in a model – the oracle), with the actual behavior of the system while running. Task models were used to generate the oracle, as they describe the intended use of the system. The possibility of introducing mutations in the task model was also explored, so that deviation from the norm (for example, user errors) could be considered during the analysis [2]. The need for an oracle, however, means that a model is still needed to perform the analysis.

CURRENT AND FUTURE WORK

While the approaches described above have proven able to provide insights into the design and trustworthiness of interactive computing systems, the need for models presents a barrier to adoption. Lighter-weight alternatives are needed when the cost of the modelling step in not justifiable.

We have been exploring alternatives to reason about the quality of an interactive computing system directly from its implementation. The goal is to better integrate the analysis with development contexts more centered around the production of code, such as some agile approaches or less structured development processes, as is sometimes the case on Web and mobile applications' development.

One first approach was to use static analysis techniques to reverse engineer models from source code [7]. This allowed us, not only to apply the model analysis techniques we already had available on the outcome of the reverse engineering step, but also to identify problems directly in the code during the reverse engineering step itself. For example, user interface components declared but never used. This type of approach, however, is hard to generalize, which, especially in the case of Web applications, becomes a problem given the multiplicity of options regarding implementation technologies. It is also not easy to apply when we consider adaptive or dynamically generated user interfaces, as the concrete interface is only known at run time, and typically not easy to deduce from the source code alone.

More recently, we have started looking at alternative approaches to analyzing the quality of the implementation. One direction explores the know how obtained with the reverse engineering and model based testing work. It consists in applying hybrid analysis techniques to perform both direct analysis and reverse engineering [14]. The approach, targeted at Web applications, combines dynamic exploration of the user interface, with static analysis of the event listeners. This approach presents a number of benefits. Compared with dynamic analysis, it enables us to achieve better coverage of the system's state space during exploration, as well as a more detailed model of the system. Compared with static analysis, it enables us to significantly reduce generalization problems, and problems with dynamically generated user interfaces, as we analyze and extract code from the running application, limiting the amount of static analysis to a minimum. Given the distributed nature of Web applications, and to minimize generalization problems, code instrumentation is used to simulate different responses from the business logic.

Another direction is exploring the idea of code smells [9]. A *code smell* highlights some feature of the code that, while not necessarily an error, might indicate a weakness in the system's implementation. Our ultimate goal is to apply the concept to user interfaces analysis and define a set of usability related smells. As a starting point we are looking at traditional WIMP interfaces, but we envisage that for different interaction techniques different smells will have to be defined. Thus far, we have found that while some of the smells we have identified relate to the implementation's quality, others relate to the quality of the resulting user interface. How to automate their analysis is still an open issue. For code related smells, the know how on reverse engineering can once again be leveraged. For user interface related smells, we intend to explore which type of models might be needed (and possible to obtain) in order to support their detection.

An alternative to attempting to avoid the need for models in the analysis, is to improve their added value. If more value can be obtained from the modelling process, and if its cost can be lowered, then the cost of developing a model can be better justified. APEX already points in that direction by making models the basis for both formal verification and prototyping. We are currently extending those ideas into IVY, exploring

the feasibility of using MAL models to support the prototyping of the user interfaces in the style of [12].

A related but somewhat different role is performed by models' animation. While a prototype helps validate the design with users by presenting them with a version of the user interface, which is derived from (and controlled by) the model, an animation is intended to help in an initial validation of whether the model is the intended one. This is achieved by supporting direct interaction with the model itself. In the case of APEX this is supported by CPN Tools. Regarding IVY, however, model validation is currently carried out by exploring the design through proving properties. This is an expensive process that can be made more cost effective by supporting direct interaction with the model. Once some degree of confidence about the model is achieved via animation, further analysis can then be carried out through verification. Bringing together model animation, formal verification and prototyping, all based in a single model will considerable raise the cost effectiveness of building the model.

CONCLUSION

The pervasiveness of software makes us more and more dependent on its quality. It is thus unfortunate that for the most part the quality of the software being produced is still somewhat lacking. This is also true, and particularly relevant, of interactive computing systems. If their development is to be considered an engineering discipline, then we need methods and tools to help us reason about, and predict, the quality of systems from early in the design process.

This paper has provided a brief overview of the work we have been carrying out under the generic umbrella of reasoning about, and ensuring the quality of, interactive computing systems. The described techniques and tools should be seen as an addition to the toolbox of already existing techniques and tools available for interactive computing systems development. Some ideas currently under development have also been highlighted.

ACKNOWLEDGMENTS

IVY development is currently funded by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT) within the LATiCES project (NORTE-07-0124-FEDER-000062).

The APEX project is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-015095.

Work on model based testing is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-020554.

REFERENCES

1. Anderson, J. R. *How can the human mind occur in the physical universe?* Oxford University Press, New York, NY, USA, 2007.
2. Barbosa, A., Paiva, A. C. R., and Campos, J. C. Test case generation from mutated task models. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, F. Paternò, K. Luyten, F. Maurer, P. Dewan, and C. Santoro, Eds., ACM (2011), 175–184. ISBN: 978-1-4503-0778-9.
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting with Computers 15*, 3 (2003), 289–308.
4. Campos, J., Doherty, G., and Harrison, M. Analysing interactive devices based on information resource constraints. *International Journal of Human-Computer Studies 72*, 3 (March 2014), 284–297.
5. Campos, J., and Harrison, M. Modelling and analysing the interactive behaviour of an infusion pump. *Electronic Communications of the EASST 45: Formal Methods for Interactive Systems 2011* (2011). ISSN: 1863-2122.
6. Campos, J. C., and Harrison, M. D. Interaction engineering using the ivy tool. In *ACM Symposium on Engineering Interactive Computing Systems (EICS 2009)*, ACM (New York, NY, USA, 2009), 35–44.
7. Campos, J. C., Saraiva, J., Silva, C., and Silva, J. C. GUIsurfer: A reverse engineering framework for user interface software. In *Reverse Engineering - Recent Advances and Applications*, A. Telea, Ed. InTech, 2012, ch. 2, 31–54.
8. Clarke, Jr., E. M., Grumberg, O., and Peled, D. A. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.
9. Fowler, M. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
10. International Organization for Standardization. ISO 9241-210:2010 Ergonomics of human-system interaction – part 210: Human-centred design for interactive systems, 2010.
11. Memmel, T., Gundelsweiler, F., and Reiterer, H. Agile human-centered software engineering. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 1*, BCS-HCI '07, British Computer Society (Swinton, UK, UK, 2007), 167–175.
12. Oladimeji, P., Masci, P., Curzon, P., and Thimbleby, H. PVSio-web: a tool for rapid prototyping device user interfaces in PVS. In *Proceedings of the 5th International Workshop on Formal Methods for Interactive Systems (FMIS 2013)* (2013).

13. Ratzer, A. V., Wells, L., Lassen, H. M., Laursen, M., Qvortrup, J. F., Stissing, M. S., Westergaard, M., Christensen, S., and Jensen, K. CPN Tools for editing, simulating, and analysing coloured Petri nets. In *Proceedings of the 24th international conference on Applications and theory of Petri nets*, ICATPN'03, Springer-Verlag (Berlin, Heidelberg, 2003), 450–462.

14. Silva, C. E., and Campos, J. C. Combining static and dynamic analysis for the reverse engineering of web applications. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2013)*, P. Forbrig, P. Dewan, M. Harrison, K. Luyten, C. Santoro, and S. Barbosa, Eds., ACM (2013), 107–112.

15. Silva, J., Campos, J., and Harrison, M. Formal analysis of ubiquitous computing environments through the APEX framework. In *ACM Symposium on Engineering Interactive Computing Systems (EICS2012)*, ACM (2012), 131–140.

16. Silva, J., Campos, J., and Harrison, M. Prototyping and analysing ubiquitous computing environments using multiple layers. *International Journal of Human-Computer Studies 72*, 5 (May 2014), 488–506.

17. Silva, J. L., Campos, J. C., and Paiva, A. Model-based user interface testing with spec explorer and concurtasktrees. *Electronic Notes in Theoretical Computer Science 208: 2nd International Workshop on Formal Methods for Interactive Systems (FMIS 2007)* (2008), 77–93.

18. Sousa, M., Campos, J., Alves, M., and Harrison, M. Formal verification of safety-critical user interfaces: a space system case study. In *Formal Verification and Modeling in Human Machine Systems: Papers from the AAAI Spring Symposium*, AAAI Press (2014), 62–67.

19. Young, R. M., Green, T. R. G., and Simon, T. Programmable user models for predictive evaluation of interface designs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '89, ACM (New York, NY, USA, 1989), 15–19.

The Role of Semantic Data in Engineering Interactive Systems

Jürgen Ziegler and Timo Stegemann
University of Duisburg-Essen, Interactive Systems Research Group

47048 Duisburg, Germany
{juergen.ziegler, timo.stegemann}@uni-due.de

ABSTRACT
Semantic technologies, as advocated and used in the context of the Semantic Web, are increasingly seen not only as a pre-requisite for the automated processing of distributed data on the Web, but also as a basis for new methods and tools for engineering interactive systems. In this position paper, we briefly review the state of the art related to visualizing and exploring semantic data, in particular Linked Open Data, as well as techniques for developing user interfaces for semantic data. We also refer to some of our own work in this field. Based on this overview, we present a range of open research issues that we organize around three paradigmatic concepts that might inform and guide future developments in this area. The three paradigms discussed in this paper are: constructive exploration, dynamic contextualization, and transparent integration of semantic data. These conceptual contributions together with the concrete developments described shall serve as input to a broader discussion of a roadmap for the future engineering of interactive systems.

Keywords
Semantic Web, visual exploration, context-adaptation, semantic widgets

INTRODUCTION
Since the first articulation of the vision of a Semantic Web (Berners-Lee et al., 2001), the amount of data on the Web represented with semantic techniques has increased enormously. In contrast to the conventional Web which links web pages as complete documents, the Semantic Web links data based on a uniform data model. This model consists of elementary statements in 'subject-predicate-object' form, expressed with RDF, and uses shared vocabularies defined as ontologies by means of RDFS and OWL. Linking the individual statements results in single, huge data graph, which is often referred to as the Web of Data.

Today, large amounts of semantic data are available openly on the Web, often automatically extracted from conventional information sources such as encyclopedias (e.g. DBPedia which is extracted from Wikipedia, geographical databases, bibliographies, product data and many others (for an overview see, e.g. Bizer et al.,2008). The LOD (Linking Open Data) initiative has invested considerable effort in interconnecting different datasets, resulting in the LOD Cloud which has meanwhile reached an enormous size. Beyond general public information sources, semantic techniques are also making their way into more specific application domains such as electronic commerce (e. g., Hepp, 2008) or corporate document management (e.g., in the current Microsoft Sharepoint).

While the original objectives of semantic techniques were mainly directed at making Web information machine-processable, it is increasingly recognized that linked open data and other semantic data pools are valuable information sources that can be used interactively by end users. This creates a need for methods and tools that allow users to interact with Semantic Web data directly, rather than through a web application that integrates and delivers data in standard web pages.

In contrast to conventional Web front ends, user interfaces and search tools for Semantic Web data can exploit the semantics of the underlying data structure, for instance, to let users explore the data under different perspectives and formulate more targeted and complex queries. The potential of interactively exploring semantic data has been demonstrated by a variety of tools that use the different semantic relations for browsing the data or for creating search facets that allow users to flexibly filter the data. Visual techniques for formulating complex queries have been developed, for instance, in prior work of ours (Heim, Ertl, & Ziegler, 2010; Heim et al., 2009). In addition to providing improved search and exploration capabilities, however, semantic representations can also enable users to construct their own views of a semantic data pool, which can either be used for exploring the data in some user-defined visual configuration or for presenting (and potentially editing) a more or less complex cut-out of the large RDF based data graph.

Semantic techniques open up new approaches to end-user tailoring or even development of interactive applications. Due to the explicit and user-inspectable representation of semantic models in the form of ontologies, to which the (instance) data are directly linked, a range of new development methods and tools can be conceived which also have implications for the architecture of interactive systems. In this position paper, we describe some of the new conceptual aspects of engineering interactive systems based on semantic data which may change the

way interactive systems and user interfaces will be developed in the future. In the following, we first present a brief review of the state of the art and describe some of our own developments. We then discuss three lines of research in which we believe semantic techniques will have an impact on engineering interactive systems in the future.

RELATED WORK

A variety of tools have been developed in recent years that allow users to browse Semantic Web data directly. Well-known examples of Semantic Web Browsers include Tabulator (Berners-Lee et al., 2008), Parallax (Huynh & Karger, 2009), and Humboldt (Kobilarov & Dickinson, 2008). Different models and metaphors have been proposed which can roughly be subdivided into page-based and graph-based techniques. Page-based approaches show a resource as a single Web page with links representing the relations to other resources while graph-based approaches provide a visualization of the RDF graph directly. RDF data also lend themselves well to faceted browsing since related concepts can be used to filter the instance set of some target concept. This technique has been used in a number of systems in which facets were mainly represented as menus or links on pages showing the filtered results (as an example, see schraefel et al, 2005).

In our own work, we have developed several tools for visualizing and exploring RDF data. *Facet Graphs* combine the benefits of graph-based RDF visualizations with facetted browsing and allow users to formulate complex facetted queries in a visual form (Heim et al. 2010). In this tool, a graph visualization was extended to contain set-valued nodes, showing all instances belonging to the concept represented by the node. This allows users to formulate complex queries by selecting filter attributes from adjacent as well as distant concepts. *RelFinder* is a visual tool for detecting so far unknown relationship or patterns. RelFinder (Heim et al. 2009) is a browser-based system that can take two or more elements of an RDF dataset as starting points for extracting the relations among those elements and visualizing them as a network that can further be manipulated and explored. A marketing manager, for example, could thus identify products of competitors in a certain product category, find out whether they were used for similar purposes, or check whether they use the same suppliers. RelFinder can therefore be regarded as an interactive knowledge discovery tool.

While tools for browsing semantic data are typically generic and support the unconstrained exploration of the data, in many cases more domain- and application-specific user interfaces are needed. Several techniques have been proposed to (rapidly) develop UIs for semantic data, mostly with a focus on presentation.

The Xenon project (Quan & Karger, 2005) is an example of an approach for transforming RDF data into presentations based on a stylesheets. Based on the concept of XML-Stylesheets (XSLT), the authors developed a RDF-based stylesheet language, which defines concepts for transforming RDF-data using lenses and views. The purpose of lenses is data selection from instances whereas views describe the visualization of data elements. Just like with XSLT it is possible to embed HTML markup directly into the stylesheet for generating HTML representations of the data. As a follow-up to Xenon, Fresnel (Pietriga et al., 2006) still uses lenses for selecting data but replaces the view concept by formats that define, for instance, whether the data selected should be presented as a link, an image or as text.

OWL-PL (Brophy, 2010), is a further language for transforming RDF/OWL data into (X)HTML. The language is strongly inspired by XLST and has the main goal to provide a simple transformation language for semantic data. OWL-PL allows the combination of transformational and representational markup. The language defines stylesheets, which are connected to semantic data by using a stylesheet ontology. The ontology describes how specific RDF classes are related to stylesheet elements. With the introduction of LESS [ADD10], a complete workflow from creating and processing templates for semantic data up to sharing templates between users is described. The declarative template language LeTL (LESS template language) is specified as a Smarty based templating language. It can process and transform semantic data from RDF documents or data requested by SPARQL queries.

Work in our group has also addressed techniques for facilitating the construction of frontends for semantic data. In (Stegemann et al., 2012), we describe X3S, a technique for composing presentations of semantic data that can be created by simple drag-and-drop actions. A recent development are semantic widgets which will be further described in the following section.

RESEARCH CHALLENGES

In the following, we describe selected areas which pose particular challenges for future research related to semantic data and engineering user interfaces. To characterize the challenges involved and to indicate possible directions, we group the issues around three principles which we also see as conceptual contributions to discussing future research directions: *Constructive Exploration*, *Dynamic Contextualization* and *Transparent Integration of Semantic Data* which will be discussed in the following sections.

Constructive Exploration

Semantic data sources, in particular linked open data, are represented by very large graph structures, typically comprising billions of individual statements (e.g. about 30 billion triples in LOD in 2011). In contrast to the conventional Web where complete documents are retrieved, the situation is different for semantic data. Users searching and exploring the data need find suitable starting points for their exploration and need to exploit the semantic relations to link them to related information.

Making sense of the data often requires filtering or aggregating them from different perspectives to extract those parts of the data graph that are relevant to the question at hand. Essentially, this means that users must be provided with tools that allow them to construct individual, potentially complex views on the data that can be used for further exploration. Due to the close integration of searching, exploring and visualizing the data, we call this process "constructive exploration". An example of this approach are the aforementioned FacetGraphs where users construct an arbitrarily complex graph of concept nodes that contain lists of instances of the respective concept. Items can then be filtered over direct or even indirect connections to other concepts, allowing the user to change ("pivot") the direction of filtering or apply several filters simultaneously.

We see this as an example of a much wider range of possibilities where users can compose and configure visualizations or user interfaces in general to suit their individual task and information needs with respect to semantic data resources. Future research will need to address suitable UI metaphors, interaction tools and implementation approaches to create such flexible and user-driven visual environments for exploring semantic data.

Dynamic Contextualization

In context-adaptive systems and user interfaces, there still exists a considerable gap between theoretical considerations and actual technical solutions. In one of the most frequently cited papers on context, Dey and Abowd (2000) argue that any information that characterizes the situation of some entity can be considered as context. However, concrete examples of adaptive systems mostly use a much more restricted view, restricting context to a priori defined factors such as location, time or the device used. Broad notions of context are hard to operationalize. To reduce these limitations, we argue that context-adaptivity should be considered as a process of *dynamic contextualization* in which the system offer means for determining relevant context on the fly when the decision about some adaptation is about to take place (Hussein et al. 2014).

Semantic models and data can play an important role in achieving such dynamic contextualization processes. Dynamic contextualization can be operationalized by a process that comprises the following steps: 1. Representing long-term user information and knowledge about the domain and potential context factors as semantic models and data. 2. Sensing potentially relevant information about the user's current state. 3. Dynamically identifying contextually relevant elements, e.g. querying the semantic database 4. Reasoning (for instance inferring recommendations) based on the extracted context. These four steps roughly correspond to a four layer model proposed in previous work of one of us (Haake et al., 2010), where a semantic model represents the knowledge base used for subsequent contextualization steps. In this framework, semantic models provide a solid and comprehensive basis for adapting UIs, using the reasoning mechanism most suitable for the current adaptation problem.

While we have explored this approach successfully in relation to context-aware hybrid recommenders and consider it promising, there are still many open research issues. Open questions exist, in particular, with respect to processing and abstracting sensed context, as well as to integrating external, sensor-based context with user models that capture behaviour or preferences. Finally, suitable reasoning mechanisms as well as meaningful and usable adaptations are required .

Transparent Integration of Semantic Data

While interacting with semantic data directly offers many opportunities for the targeted retrieval or exploration of facts and relations, in many cases it is desirable to integrate semantic data with conventional Web content to create more dedicated, application-specific user interfaces. Although this has been the subject of some research, there are still obstacles that prevent developers from making use of this option. There are issues related to architectural aspects as well as to the tools and languages used for doing so. Furthermore, there are currently no tools that would be usable by end users for integrating semantic data, e. g., in low-threshold publishing tools such as blogs or wikis.

To overcome some of these obstacles, we propose the concept of Semantic Widgets that shield users from the complexity of the RDF query language (SPARQL3) and facilitate the construction of Web applications that use and integrate semantic data. To realize the concept, we have been working on SemwidgJS (Stegemann & Ziegler, submitted), a JavaScript library for displaying Linked Open Data through Semantic Widgets. SemwidgJS can be integrated in almost any standard HTML webpage and handles the querying, processing and displaying of semantic data. While existing libraries sharing a similar goal only comprise widgets for information visualization purposes, SemwidgJS also features widgets for typical UI elements such as labels, links, and text input fields. To make querying semantic data easy, SemwidgJS supports its own simplified, path-based query language – SemwidgQL which is used as an alternative to standard SPARQL. We envisage that by providing suitable tools, users can be enabled to define the queries handled by the widgets in a completely visual way, combining exploration of the data with the definition of (reusable) widgets that can be embedded in standard Web pages.

Beyond this specific example, we see a range of research issues that need to be solved to bring semantic techniques closer into the hands of end users. In terms of architecture, server-side and client-side techniques need to be further investigated. SemwidgJS widgets, for example, are completely processed in the client, Web

authors therefore only need to include suitable widget mark-up in their HTML code without the need to have specific functionality available on the server side. Integrating semantic Web services in an easy, user-definable manner is also an area where more research will be needed to move beyond presentation-oriented applications and to make mash-ups of Web content and semantic data fully interactive. Eventually, this requirement may change the overall architecture of interactive applications, possibly resulting in new solutions beyond current practices of service-oriented systems.

CONCLUSIONS

Semantic Web techniques offer a high potential for new UI styles and capabilities, and for changing the way interactive systems are designed and built. Yet, many research challenges still lie along this way. Paradigms and metaphors for interacting with semantic data, for instance must be further investigated to increase usability. More tightly integrating semantic user and context models could lead to more effective and transparent adaptations. Finally, we believe that semantic techniques could empower end users to define and compose (without programming) their own personalized applications and user interfaces.

REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. Scientific American, (May 2001).

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'ommeaux, E. and Schraefel, m.c. Tabulator Redux: Browsing and writing Linked Data. In Proc. WWW 2008 Workshop: LDOW, (2008).

Bizer, C., Heath, T., Kingsley, I., & Berners-Lee, T. (2008). Linked data on the Web. In Proceedings WWW 2008, pp. 1265-1266.

Brophy, M. OWL-PL (2010). A Presentation Language for Displaying Semantic Data on the Web. Master's thesis, Lehigh University.

Dey, A. K., & Abowd, G. D. (2000). Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness. The Hague, Netherlands: ACM Press.

Haake, J., Hussein, T., Joop, B., Veiel, D., & Ziegler, J. (2010). Context modeling for adaptive collaboration. Int. Journal of Cooperative Information Systems, 19, 71-120.

Heim, P., Ertl, T., & Ziegler, J. (2010). Facet Graphs: Complex semantic querying made easy. In Proceedings ESWC 2010. Berlin: Springer.

Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., & Stegemann, T. (2009). RelFinder: Revealing Relationships in RDF Knowledge Bases. In Proceedings. 3rd Int. Conf. on Semantic and Media Technologies (SAMT). Berlin: Springer.

Hussein, T., Linder, T., Gaulke, W., & Ziegler, J. (2009). Context-aware recommendations on rails. In CARS '09- Workshop on Context-Aware Recommender Systems (in conj. with 3rd ACM Conf. Recommender Systems).

Hussein, T., Linder, T., Gaulke, W., & Ziegler, J. (2014). Hybreed: A software framework for developing context-aware hybrid recommender systems. User Modeling and User-Adapted Interaction, 24(1-2), 121–174.

Huynh, D. and Karger, D. (2009). Parallax and companion: Set-based browsing for the Data Web. Proceedings WWW 2009.

Kobilarov, G. and Dickinson I. Humboldt: Exploring Linked Data. In: Proc. WWW 2008 Workshop: LDOW, (2008).

Pietriga, E., Bizer, C., Karger, D. R., & Lee, R. (2006). Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In ISWC '06: Proceedings of the 5th International Semantic Web Conference (Vol. 4273, S. 158–171). Springer.

Stegemann, T., Ziegler, J., Hussein, T., & Gaulke, W. (2012). Interactive construction of semantic widgets for visualizing Semantic Web data. Proceedings 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2012), Copenhagen.

Stegemann, T., & Ziegler, J. (2014, to appear). SemwidgJS: A semantic widget library for the rapid development of user interfaces for linked open data. In AST 2014 - 8th International Workshop on Applications of Semantic Technologies (in conjunction with Informatik 2014, Sept. 22-26, Stuttgart). Berlin, Heidelberg: Springer.

What programming languages for interactive systems designers?

Stéphane Chatty
Université de Toulouse - ENAC
7 avenue E. Belin, 31055 Toulouse, France
chatty@enac.fr

Stéphane Conversy
Université de Toulouse - ENAC
7 avenue E. Belin, 31055 Toulouse, France
conversy@enac.fr

ABSTRACT
We highlight the role of programming in the engineering of interactive systems, in the long term perspective of creating general theories of interaction to support engineers. We outline a research roadmap aimed at both providing designers with appropriate programming languages and understanding the nature of interactive programs.

Author Keywords
interactive software, programming languages, notations, engineering, design, theory

ACM Classification Keywords
H.5.2 Information Interfaces and presentation: User Interfaces; D.3.3 Programming Languages: Language Constructs and Features

INTRODUCTION

For the computing industry and the world in general, this has been the decade of interactive systems. Long announced by researchers, natural user interfaces have reached the industrial stage and found their way to our pockets and our bedside tables. This has changed the computing industry by giving a more central role to individual programmers and to design professionals. This also has bolstered the public awareness about software programming, with the popularization of sentences such as "programming is the new literacy" or "program or be programmed" [26].

However, there is a paradox: what makes computers so interesting is their interactivity, and still the programming techniques that are proposed to the eager masses are the old techniques, invented for designing algorithms and not interaction! This might create some disillusionment with computer science, and we indeed are observing its first signs in engineering students. We contend not only that addressing this paradox is our responsibility, but that creating programming languages for interactivity would bring benefits to engineers.

In this article we discuss three statements and their consequences for research on engineering interactive systems:

- *engineers need general theories of interaction*
- *designing interactive systems is programming*
- *programming languages are user interfaces.*

We elaborate on these statements and why they should be important to our research community, then we outline possible areas of research aimed at exploring their consequences.

WE NEED GENERAL THEORIES OF INTERACTION

Engineering has been defined by the Engineers Council for Professional Development, in the United States, as [1, emphasis ours]:

> The creative application of *scientific principles* to *design or develop* structures, machines, apparatus, or manufacturing processes [...] *or to forecast their behaviour* under specific operating conditions [...].

Let us first note the central role of scientific principles in this definition: engineers need scientific theories, and progress in engineering is often triggered by theoretical progress. What is less intuitive is the combination of roles assigned to engineers: to design (with an emphasis on creativity) or to forecast. We derive two lessons from this:

- designing is a major component of engineering
- scientific theories are used both to design and to forecast

Indeed, examples abound of theories that are used to both design and forecast. Mechanical engineers use the same concepts of forces and pressures to forecast the behavior of bedrock and to design a bridge built on it. The same holds for chemical engineering, for bio-engineering, and even for traditional software engineering. Theories of computation allow to describe computation systems (even natural ones) and to predict their behavior. Through programming languages derived from them, they also allow to design computation software and to forecast its behavior. What allows these theories to support both design and forecast activities is their generality: they encompass all the relevant aspects of the system being designed and its environment. We contend that this should be an explicit goal in the field of interactive systems too.

Engineering interactive systems involves, before designing and developing a software system, the analysis of the human and physical environment in which it will operate. This requires the ability to model software, human cognition and perception, activities and tasks, application domains, and even physical interaction. Currently, there are specialized theories for each of these, and we spend considerable efforts developing empirical methods that help engineers combine

these theories. But this focus on methods should only last as long as we feel compelled to teach separate theories in our engineering courses. The ultimate goal should remain the elicitation of general theories that encompass all these aspects, as attempted for instance by Palanque et al. when using Petri nets to model the behavior of both the user and the software [23].

DESIGNING INTERACTIVE SYSTEMS IS PROGRAMMING

Programs are a central concept in theories of computation, and in the whole field of software engineering. At the opposite, in the field of interactive software engineering they are often relegated to the status of mere by-products. But programs are nevertheless a major part of interactive systems, and any general theory of interaction will need to clarify their status and the role of programming. This section aims at demonstrating that the activity of designing interactive systems can be considered as programming.

Generalized programming

In the following, we consider programs as descriptions of the behavior of an entity when an execution device runs them, and programming as an act performed by a human to design the description. It is possible to picture the activity of programming as carried out by professionals trained in computer science and software engineering, who write algorithms in C, Java or C#. We contend that this picture is harmful to our ability to support engineers who design or need to forecast the behavior of interactive systems. Both the design of systems, that is the definition of their structure and their behavior, and the analysis of their environment are closer to programming than it seems.

To start with, programming does not only consist in writing. It also involves reading, understanding, checking, designing, forecasting the behavior of code. It is an engineering activity, and cannot be distinguished from the engineering of interactive systems on purely methodological grounds.

Then programming has ceased to be a task reserved to computer scientists and software engineers. More and more graphical designers, web designers and communication professionals learn programming languages for producing parts of interactive systems [8, 21]. For them, programming is part of their activity just like drawing sketches. Languages and environments such as Processing [25] have been created explicitly for designers.

Programming should also not be restricted to the engineering of algorithms. When designing an interactive system, a number of entities that must be analyzed or designed can be modelled as programs. The most obvious is the interactive behavior of visual components: it is now commonplace that interaction designers program them, using whatever notation is available to them. The same holds for animation, and even for graphics themselves: sometimes, graphical designers want to produce effects that are best described as programs [8]. Operational procedures, often present in safety critical systems, are programs. Even user tasks are similar to programs, as illustrated by the conceptual similitude between CTT [20] operators and parallel programming languages control structures.

Programming algorithms, programming interaction

Overall, designing interactive systems and programming are much closer than usually advertised. We suggest that the traditional view of programming is biased. Turing and the generations that came after him have created such a consistent body of theories and programming languages that the theory of computation is used ubiquitously for analyzing systems, for designing algorithms, and even as a natural science [3]. This success sometimes obscures the existence (even the prevalence!) of other kinds of programs.

In this context, two courses of actions are possible for researchers. The first option is to design languages that help user interface designers assimilate concepts from the theory of computation. The second option, that we suggest is more promising, would be to acknowledge the difference and design adequate languages and theories to support user interface designers. Then, over the years, reaching the ability to consider interactive programs, human procedures and tasks as manifestations of the same theoretical principles could lead to a more balanced situation, with two kinds of programs and two bodies of knowledge: algorithms, and interaction.

In the rest of this paper we focus on the second option: how can we design programming languages and theories for all actors of user interface design? We also abandon any distinction between “user interface programming” and “programming in general”, because in today’s computing industry most programmers are confronted to user interfaces. Therefore, we choose to tackle the following question: *how can we design theories, notations, languages and tools that support future programmers, those who will have various backgrounds and will all build interactive software?*

PROGRAMMING LANGUAGES ARE USER INTERFACES

Software programming is an act performed by a user through a machine. As such, it is like any other computer-supported activity and requires usable tools, i.e. tools that enable their users to accomplish a task with a minimum amount of resources and in a delightful way. A review and classification of the usability requirements expected of interactive development tools has been made in [16].

When thinking about supporting software programmers, integrated development environments (IDE), user interface management systems (UIMS) and user interface builders are the first tools that come to mind. But programming languages play a more central role because they ultimately condition how programmers think about programs. To analyze the role of programming languages, and more generally, of any notation, we use Norman’s theory of action [22] and study three aspects of interaction with them: evaluation, conceptual model and execution.

Evaluation

Programming languages allow programmers to express programs through a notation. Whether textual or so-called “visual”, notations employ various graphical “features”: texts,

shapes, alignments, colors, arrows, etc, to encode information. The representation of a program is called "the code".

An implicit but important aspect of programming languages is that they must support the production of readable code, for oneself and for others [24]: "Programs must be written for people to read, and only incidentally for machines to execute [2]". In this regard, the readbility of code is like the readbility of any visual representation: the first step toward forming a mental model and acting.

The graphical appearance of the code has been shown to have an impact on understanding. For example, indentation length has been experimentally shown to have an impact on the comprehension of code: 2- and 4-space indentation makes readers better at understanding the code than 6-space indentation, for both novice and expert readers [18]. More surprisingly, Green et al. found that textual representations outperformed LabView's graphical representations for each and every subject [12].

Conceptual model

Interacting with a tool is more than just its look and feel. One of the essential aspects is the underlying conceptual model, that is an explanation, usually highly simplified, of how a system works [22]. For example, the conceptual model of a file system relies on the concepts of File and Directory and the related operations. It can be represented either as icons, or as lists of names in a command line user interface. Conceptual models are considered essential to usability by HCI specialists: a badly designed conceptual model is often at the root of poor usability.

Programming languages are no exception to this rule: they can be analyzed as a visual representation that reflects an underlying conceptual model. For example, LISP code with its parentheses is one possible representation of the underlying hierarchy of expressions. Another representation of the same program would be a graphical tree that shows the hierarchy in 2D.

A conceptual model shapes the way their users think about their problem at hand and the ways to solve it. In the case of programming languages, it must be usable enough to help programmers think about, design, write and read programs. The conceptual model of a programming language is often derived from a general theory. Consequently, it does not only support the production of code, but the analysis of programs and their environment, to the extent of what the theory can describe. For example, the "functional" conceptual model[15] is well-adapted to the description of computation, while the "reactive" conceptual model is well-adapted to interactive behaviors. This brings us back to the aforementioned general theory of interaction: to produce usable programming languages for interactive systems designers, theories that encompass the appropriate concerns must be available.

Execution

When programming, execution consists in writing code or modifying it. This involves actions such as creating entities and referring to existing entities. IDEs are often considered as instrumentation of these tasks, designed to make programming with a given language more usable. For example, refactoring tools in current IDEs such as Eclipse enable programmers using functional or object-oriented languages to efficiently modify the names of functions or object methods. But, prior to IDEs, the evolution of languages can also be considered as a process to offer better support for these actions. For example, method inheritance is a way to factor common code in a single place, thus facilitating the evolution of behavior in multiple parts of the code ("mass updating" [13]). Similarly, aspects offer programmers the possibility to express cross-cutting concerns in a single place.

Moreover, some of the properties associated to "good software" can be related to usability concerns. For example, the goal of modularity is related to action and interaction: it is supposed to facilitate the maintenance of code since with well-modularized software a modification of a component performed by a programmer requires minimal adaptation and rewriting on other components.

RESEARCH DIRECTIONS

Our three statements and their discussion can be translated in a long term goal: provide interactive systems engineers with usable languages and notations for designing, developing and analyzing systems, grounded in theories that they can apply to forecast their behavior. How can this be turned into practicable research directions? The state of the art in the extended field of user interface engineering, as well as the history of traditional computer science, provide many possibilities. We list a few here:

Eliciting functionality

Programming language designers have spent decades to identify the functions of programming languages that best support traditional programmers, alone or in groups. Which of these functions are relevant for interactive systems designers, what requirements are not covered and how can they be addressed? Some authors have started to address this question [8, 21, 16] but this is only a start.

Conceptual unification

A number of concepts have been proposed to describe the behavior of interactive systems. The design of programming languages, as much as the design of theories, traditionally requires that the relationships between concepts are defined. This usually involves the definition of primitive concepts from which other concepts are derived. Such unification has been attempted in the past [5, 14] and should be pursued, even if this implies reaching out to disciplines that are currently alien to our field, including philosophy. The work presented in [17] is an example in this direction.

Formal definition of concepts

Little effort is devoted in our domain to establish consensual definitions of concepts such as task, activity, event, component, interface, animation, etc. Actually, it would be difficult to negotiate any solid consensus without more formal

candidate definitions: it is easy to agree with different interpretations in mind. Progress toward more general theories could include collective work on formal definitions, using standard community tools such as workshops, incremental or controversial publications, and consensus statements. For instance, the word "specification" has grown a particular meaning in software engineering, sometimes very different from its meanings in other engineering fields. Must we adhere to this idiomatic meaning and why? An article written in French by one of the authors [6] has sparked verbal debate about this, and it might be useful to have more structured debates, anchored in activity analysis of engineering.

Designing language concepts

If we consider that interactive systems programming is not well supported enough by existing programming languages, we should design new ones: programming languages for professional programmers and for interaction designers, notations for analysis, etc. There already are many research efforts in this direction. However, the goal of producing general theories that encompass all engineering activities requires that the concepts used by all languages be compatible, even if the notations are different. Our own research suggests that the conceptual models of traditional programming languages are not sufficient to fully describe interactions and that new, more comprehensive and unifying models of execution should be used instead. More research should be conducted to assess this aspect of the usability of programming languages, in order to identify the relevant properties and to design appropriate evaluation methods.

Designing language notations

We have already started to study the design of language notations. We notably have analyzed and modeled the process of perceiving a program using a framework based on the Semiotics of Graphics [9]. This work shows that code representation is not about aesthetics but performance, and should not be an art [11] but a science following principles from visual perception. It also suggests that there may be no substantial difference in terms of graphical perception between textual and visual languages. The Physics of Notations framework focuses on the properties of notations [19]. It adresses numerous aspects of graphical properties and defines several principles for the design of notations e.g., semiotic clarity, perceptual discriminability, semantic transparency, visual expressiveness, etc. In addition, designing a programming language should use a "programmer-centered design" approach: it should emphasize the act of designing representations targeted at tasks meaningful for end-programmers, and not designing the representation in isolation. Such work should be of interest for software engineers who often use various UML diagrams (another notation) to document their software.

Consolidating available results

A number of available results in user interface engineering have limited impact or are only used as general guidelines because they cannot be used directly by programmers and designers: architecture patterns, language constructs implemented in toolkits, dedicated algorithms, etc. A requirement for the design of new languages should be that these results can be checked against the proposed languages, so that they can be reused more directly. Reciprocally, effort should be spent on identifying available results and assessing against the proposed designs. For instance we have carried out an assessment of software adaptation against reactive programming [17] and it would now be useful to determine how architecture patterns proposed for plasticity translate in this framework. As another example, we are working on how the MDPC pattern [10] fits in an interaction-oriented language. Interestingly, some of the available results currently are implemented in operating systems [4, 7] and this is reminiscent of the relationships that existed in the past between new languages and new operating systems.

CONCLUSION

In this article, we have highlighted the role of programming in the engineering of interactive systems, in the long term perpective of creating general theories of interaction to support engineers. We have outlined a research roadmap aimed at both providing designers with appropriate programming languages and understanding the nature of interactive programs.

As researchers on engineering, one of our roles is to provide engineers with better theoretical tools, including languages and notations. As researchers on human-computer interaction, we have tools and methods that no other scientific community has for designing new theories, languages and notations. What about eating our own dog food?

REFERENCES

1. *Encyclopaedia Britannica*, vol. 8. 1972.
2. Abelson, H., and Sussman, G. J. *Structure and Interpretation of Computer Programs*, 2nd ed. MIT Press, Cambridge, MA, USA, 1996.
3. Arrighi, P., and Dowek, G. The physical Church-Turing thesis and the principles of quantum theory. *International Journal of Foundations of Computer Science 23*, 5 (2012).
4. Chapuis, O., and Roussel, N. Metisse is not a 3d desktop! In *Proceedings of the ACM UIST'05 conference* (2005), 13–22.
5. Chatty, S. Extending a graphical toolkit for two-handed interaction. In *Proceedings of the ACM UIST'94 conference* (Nov. 1994), 195–204.
6. Chatty, S. Réconcilier conception d'interfaces et conception logicielle : vers la conception orientée-systèmes. In *Proceedings of Ergo-IHM 2012* (2012).
7. Chatty, S., Boulabiar, M.-I., and Tissoires, B. L'évolution de linux vers les nouvelles formes d'ordinateurs personnels. In *Proceedings of the 6th International Conference on the Sciences of Electronics, Technologies of Information and Telecommunications (SETIT 2012)* (2012).
8. Chatty, S., Sire, S., Vinot, J., Lecoanet, P., Mertz, C., and Lemort, A. Revisiting visual interface

programming: Creating GUI tools for designers and programmers. In *Proceedings of UIST'04*, Addison-Wesley (Oct. 2004), 267–276.

9. Conversy, S. Unifying textual and visual: a theoretical account of the visual perception of programming languages. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Splash Onward! '14, ACM Press (New York, NY, USA, apr 2014), (to be published).
10. Conversy, S., Barboni, E., Navarre, D., and Palanque, P. Improving modularity of interactive software with the MDPC architecture. In *Proceedings of EIS (Engineering Interactive Systems) conference 2007, joint HCSE 2007, EHCI 2007 and DSVIS 2007 conferences, Lecture Notes in Computer Science*, Springer Verlag (March 2007), 321–338.
11. Green, R., and Ledgard, H. Coding guidelines: Finding the art in the science. *Commun. ACM 54*, 12 (Dec. 2011), 57–63.
12. Green, T., and Petre, M. When visual programs are harder to read than textual programs. In *Proc. of the 6th European Conference on Cognitive Ergonomics (ECCE 6)* (1992), 167–180.
13. Green, T. R. G., Borning, A., O'Shea, T., Minoughan, M., and Smith, R. B. The Stripetalk papers: Understandability as a language design issue in object-oriented programming systems. In *Prototype-Based Programming: Concepts, Languages and Applications*. Springer, 1999, 47–62.
14. Jacob, R., Deligiannidis, L., and Morrison, S. A software model and specification language for non-WIMP user interfaces. *ACM Transactions on Computer-Human Interaction 6*, 1 (1999), 1–46.
15. Landin, P. J. The next 700 programming languages. *Commun. ACM 9*, 3 (Mar. 1966), 157–166.
16. Letondal, C., Chatty, S., Phillips, G., André, F., and Conversy, S. Usability requirements for interaction-oriented development tools. In *Proceedings of the PPIG 2010 Workshop on the Psychology of Programming* (2010), 12–26.
17. Magnaudet, M., and Chatty, S. What should adaptation mean to interactive software programmers? In *Proceedings of the ACM EICS 2014 conference* (2014).
18. Miara, R. J., Musselman, J. A., Navarro, J. A., and Shneiderman, B. Program indentation and comprehensibility. *Commun. ACM 26*, 11 (Nov. 1983), 861–867.
19. Moody, D. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng. 35*, 6 (Nov. 2009), 756–779.
20. Mori, G., Paternò, F., and Santoro, C. CTTE: Support for developing and analysing task models for interactive system design. *IEEE Transactions on Software Engineering 28*, 2 (2002), 797–813.
21. Myers, B., Park, S. Y., Nakano, Y., Mueller, G., and Ko, A. How designers design and program interactive behaviors. In *Proceedings of IEEE VLHCC'08*, IEEE Computer Society (2008), 177–184.
22. Norman, D. A. *The Design of Everyday Things*, revised and expanded edition ed. Basic Books, New York, 2013.
23. Palanque, P., Bastide, R., and Paternò, F. Formal specification as a tool for objective assessment of safety critical interactive systems. In *Proceedings of the Interact'97 conference* (1997).
24. Raymond, D. R. Reading source code. In *Proceedings of the 1991 Conference of the Centre for Advanced Studies on Collaborative Research*, CASCON '91, IBM Press (1991), 3–16.
25. Reas, C., and Fry, B. *Processing - A Programming Handbook for Visual Designers and Artists*. MIT Press, 2007.
26. Rushkoff, D. *Program or Be Programmed: Ten Commands for a Digital Age*. Soft Skull Press, 2011.