Software Design and New Media Design

Formal and Visual Tools to Design Mobile and Sensory Interfaces and Interactive Environments

Geert de Haan(✉)

Communication, Media and Information Technology, Section Media Technology/Human Centered ICT, Rotterdam University of Applied Sciences, P.O. Box 25035, 3001 HA Rotterdam, The Netherlands
geert.de.haan@upcmail.nl

Abstract. This paper discusses ETAG, a formal model for design representation, and ETAG-based design as a method for user interface design. The paper starts with an introduction of ETAG as a design representation. This is followed by a description of ETAG-based design and using the notation to represent relevant aspects of the work context. Next, we discuss the differences between computer software design and media product design, concluding that media design is a much more flexible, iterative process and prototyping-based process in which adaptation of the design of mobile applications extends into the maintenance phase. To cover further developments towards focusing on user needs and wishes by means of co-design practices, and to cover for ubiquitous computing and interaction with sensors and interactive environments, we propose to use sensory labs and to create living labs to move the usability lab into the real world.

Keywords: Formal modelling · ETAG · Software design · Media design · Design tools · Design methods

1 Introduction

In this paper we compare a formal design method for user interface design specification (ETAG; Extended Task-Action Grammar; [6, 15]) with examples of the collection of tools that are actually taught at a Human-Computer Interaction (HCI) educational curriculum in Media Technology at the bachelor level. The aim is to investigate the need for formal specification methods for user interface design; in particular to investigate the usefulness of formal modelling tools for modern cf. mobile and ubiquitous applications. Formal methods for user interface design, such as ETAG, were developed in the late eighties when the focus was on structured design methods and design for usability. Presently, as reflected in the Media Technology curriculum, the focus in teaching engineers is on designing creative applications of mobile and ubiquitous technology and services [7].

In this paper, Sect. 2 discusses Extended Task-Action Grammar (ETAG) as an example of a formal modelling approach to user interface design. As a formal modelling tool, ETAG is a fairly advanced and refined method, based on specifying what a perfectly knowing user would know about a user interface to perform tasks with it.

Section 3 discussed ETAG-based design as a design approach which uses ETAG as its main vehicle for specifying a user interface. A main element in ETAG-based design is the formal specification of task- and user interface objects, elements and commands and command-actions with a fairly restricted application of prototyping and testing.

In Sect. 4, the paper discusses the general approach to design as taught in a contemporary Human-Computer Interaction curriculum. Among the main characteristics of the Media Technology curriculum are the focus on creativity, user-centredness and the user-experience, and in the application of a loose collection of tools which each support a particular part of the design process.

Section 5 concludes the paper with a number of conclusions about the applicability of the two general approaches to user interface design. In this section, we also discuss some of the latest developments in application development, and in particular the employment of experimentation facilities such as sensor labs and living labs for concept development, design fine tuning and design evaluation. All of these developments seem to suggest that instead of relying on specifying beforehand, user interface design moves towards increasingly agile or experimentation-based approaches.

2 Extended Task-Action Grammar

ETAG (Extended Task-Action Grammar; [6, 15]) is a formal language to represent user interfaces in terms of the knowledge that a perfectly knowing user would have (in a mental model) about performing tasks. To create a psychologically valid description of user interface for design purposes, ETAG stratifies user interface knowledge into a number of levels using existential logic and written down in a formal grammar. Sowa's Existential logic [14] is used to anchor user interface knowledge in general world-knowledge. The formal grammatical notation, adopted from ETAG's predecessor, Task-Action Grammar (TAG; [12]) ensures that the description is sufficiently precise for design and implementation purposes without sacrificing psychological validity. User interface representations are stratified into levels to meet the existence of levels in human knowledge and to reflect the major decisions that occur during the design process. ETAG representations consist of a canonical basis, a user virtual machine, a dictionary of basic tasks, and a section with production rules.

The canonical basis (an ontology) lists the universally known concepts such as object, attributes and events which are used to define the specific objects, attributes, etc. of the user interface in the type specification and the type hierarchy of the user virtual machine (Fig. 1).

The user virtual machine (UVM) describes the elements and the workings of the user interface without referring to a specific implementation. Ideally, a single UVM could be used for a mobile interface, a pc-like interface, etc. In the type specification each of the concepts in the type hierarchy is defined, and additional concepts and attributes are defined to describe how the system works, as experienced by the user (Fig. 2).

```
CONCEPT ::= [OBJECT] | [PLACE] | [EVENT]
[PLACE] ::= [place.IN ([OBJECT])] | [place.ON ([OBJECT])]
[EVENT] ::= [event.KILL-ON ([OBJECT], [PLACE])]  |
  [event.MOVE-TO ([OBJECT], [PLACE])]
```

Fig. 1. A fragment of a canonical basis in ETAG for an application environment in which the user has to know that there are objects, places and events. The objects may reside on or in places, and there are events to kill or delete objects on a place and to move objects between places. This type specification fragment might apply to, for instance, a filing system, a game or a messaging system.

```
type [OBJECT > MESSAGE]
  supertype:[TEXT] ;
  themes: [HEADER], [BODY] ;
  relations:[place.ON-POS(1)([MESSAGE])] for [HEADER],
    [place.ON-POS(1)([MESSAGE])] for [BODY],
    [place.POSS-AT([MESSAGE])] for [HEADER], [BODY];
  attributes: <SENDER>, <SEND_DATE>, <STATUS>
END [message]
```

Fig. 2. A specification of a message type in ETAG which describes a message a text, consisting from one header and one message body. Each message is further characterised by a sender, a timestamp and a status attribute.

The event specification is the part of the UVM describing the workings of the system as it virtually appears to work from the point of view of the users, which may be different from how it is actually built to work. It describes what the system does when it processes the tasks that are successfully invoked by the user, using a pseudo computer program notation which describes the change in the user's task world that is described in the object specification, such as changing an attribute value or creating a new object (Fig. 3).

```
type [EVENT > COPY_MESSAGES]
  description:  for {[MESSAGE: *x]}
       [event.copy-to ([MESSAGE: *x],
       [place.ON-TAIL ([MESSAGE_FILE: *y]): *p2])] ;
  precondition: [state.IS-AT ([MESSAGE: *x],
       [place.ON-POS.(i) ([MESSAGE_FILE: *z]): *p1])] ;
  comments: "copy messages from file z onto the end of
file y"
END [COPY_MESSAGES]
```

Fig. 3. A ETAG specification of an event to copy or append messages, if there are any, from one file to the end of another file.

The dictionary of basic tasks lists the tasks which are available to the user and it links the workings of the user interface to the command specification of the tasks (Fig. 4).

```
ENTRY 6:
[TASK > COPY_MESSAGES],
[EVENT > COPY_MESSAGES]
[MESSAGE_FILE: *z]
T6 [EVENT > COPY_MESSAGES]
  [OBJECT > MESSAGE: (*x)][OBJECT > MESSAGE_FILE: *y]
comments: "copy messages from the current message file
into another file"
```

Fig. 4. An ETAG basic task or a user-level task description to copy messages to a file as the invocation of the copy-messages event along with the messages and the file as arguments.

Finally, the production rules describe, for each basic task, the command procedure in terms of the command syntax, the way of referring to command elements, the naming and labelling of command elements, and the physical actions to specify each element.

The dictionary of basic tasks and the production rules is the part of ETAG which addresses the differences between interaction styles and devices like windows systems, multitouch interaction, and interactive voice-response interfaces. To complete an ETAG specification of a system, the perceptual interface should be specified next. However, ETAG has never been extended to specify the visual aspects of the user interface, mainly because it is much easier to do graphical design by means of other tools such as paper-and-pencil or interactive user interface builders.

3 ETAG-Based User Interface Design

ETAG-based design [6] is originally developed as a design method on the basis of the ETAG notation to supplement or indeed replace software engineering design methods with one that is designed as inherently user centred. ETAG-based design is a user centred design method which guarantees or, at least, stimulates the designer to consider the user. In ETAG-Based Design user interface design is regarded as the incremental specification of the mental model of a perfectly knowing user. The design process is structured into a number of discrete steps, each covering a specific set of design decisions: task and context analysis, task design or task synthesis, conceptual user interface design, and perceptual user interface design, which consists of the design of the presentation interface and the design of the interaction language between the user and the system.

In ETAG-Based Design the ETAG notation is used to represent the analysis and design results. To this purpose, it is necessary that the notation is flexibly adapted to meet the specific purposes of the design stage. Originally, ETAG was intended only for user interface specification and not for representing the results of task analysis and task design. However, by altering the level of abstraction of the specification, the amount of detail, and the inclusion of special modelling concepts, the ETAG notation becomes useful for different purposes. For example, in modelling business procedures during task analysis, the representation is specified at a high level of abstraction without much detail, and special concepts are used to represent the decomposition of tasks and procedures and to represent agency and ownership.

ETAG-based design consists of a number of discrete phases, each with its own formal modelling specification to model task analysis results, to specify task design, concept design and user interface design, and each phase includes a particular evaluation of the specification. The phases are designed in such a way as to stimulate design iteration within each phase and to minimize the need for iterate and experiment with design options between different phases; thus enabling an easy-to-manage design process (Fig. 5).

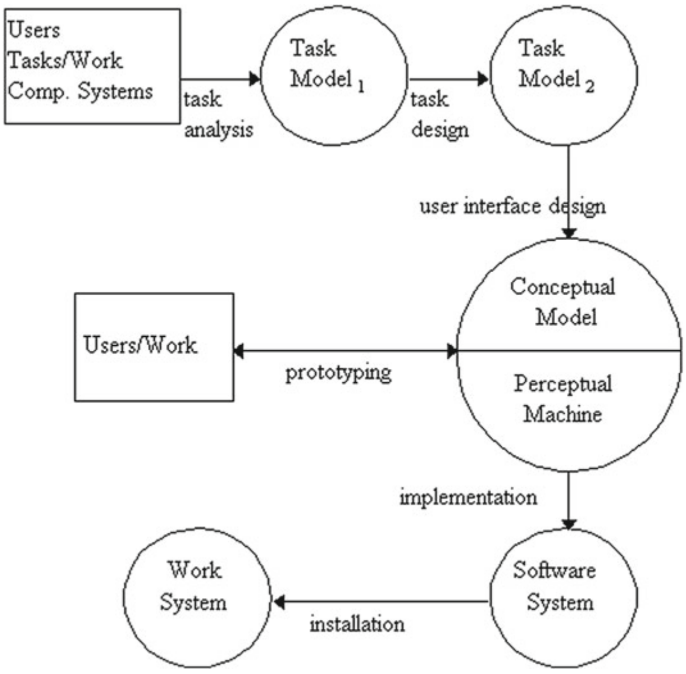

Fig. 5. The structure of ETAG-based design [6].

An advantage of using a single notation throughout design is facilitating the transitions from one design stage to the next one. Another advantage, at least in principle, is that it is easier to create tool support. This is particularly relevant for presenting ETAG as a formal model in a way which suits the background and the way of working of other stakeholders. Instead of having to translate between many different notations, in ETAG-based design, tools only need to deal with a single notation which allows for easy to automatic translation into visual specifications or programming code. This idea is similar to proposals to use a single e.g. object-oriented or XML-based representation as the underlying notation for the software engineering of user interface design specification, as proposed by Foley and Sukaviriya [5].

A final advantage of using a single notation is that designers themselves are not required to learn and use a variety of different notations. In the approaches of Paternò [11] and Constantine and Lockwood [4], for example, designers must be able to deal with a handful of different design representations, in contrast to only one notation in ETAG-based design.

ETAG was originally proposed as a competence model of the knowledge that users need to perform their tasks, much like Moran's Command Language Grammar [9] or Payne and Green's Tag Action Grammar [12] with the special addition of an ontology so that not only the translation of tasks into actions can be described but also the objects and attributes and the transitions that take place when tasks are executed. In other words, whereas TAG is able to describe what users need to know about how to delete a file, ETAG is able to describe what a file is and what happens when it is deleted.

ETAG as a formal notation is considerably harder to create than a non-formal description but it seemed worthwhile since formalisms may also be used for other purposes than mere design specification, including automatic generation of online help, generation of user interface programming specifications and code, easy calculation of usability characteristics like consistency, complexity and learnability, and the aforementioned use of a single notation throughout the whole design process (see: [6]). Not bad for a notation that is also psychologically valid, even though, at the time, the question was raised if such advantages could really counter the difficulties associated with formal representations: this might be the case for large, dependable or safety critical systems but what about the average windows utility or tablet app?

4 Designing New Media

New Media of digital media refers to all forms of purposive information transfer, carried out by digital means. Compared to analogue information carriers such as gramophone records, newspapers and paintings, digital media like web-pages and digital video, in combination with declining hardware costs and more effective tools, allow for (almost) effortless copying and editing. As a consequence of such lack of resistance to change, the design process for media products does not have to meet the same rigour of the design process for (business) software. Media products allow for a much more flexible approach to design with room for testing, experimentation, and fine-tuning of the design target with highly detailed specifications before anything can be build. In a more general way, a similar transition may be identified with every new generation of computer hardware, from the mainframe to the mini-computer, to the personal and game computer, and finally, to the smart phone and ubiquitous computing because of the technical facilities for a flexible design process in combination with increasing demand to meet market and user requirements [7].

In comparison to the design of software systems for business processes or pay rolling, the design process of new media products like interactive websites and mobile apps is lightweight, where flexibility with respect to adapting to changes in the market or the customers' wishes is a key requirement to the design and the design process. Consequently, particularly in the media area, agile design methods like Scrum [13] and Extreme Programming [2] should, at least theoretically, be most useful. In order to find out how our educational curriculum should best be adapted to the needs and wished of the companies who employ our alumni, we did a small preliminary study into design methods utilised by these companies. We asked our Media Technology students who served as their interns to name the types of product that these companies created and the main design methods used. The results indicate that virtually all of these companies either utilise scrum or comparable methods, or that they are in the process of moving from waterfall-like methods to agile design.

Because of the flexibility requirements and the lower repair costs in designing media products, the design process of media products is not only lightweight but also tends to consist of a variety of tools to suit the job without much reliance on a particular design notation. The following list contains about all user interface design representations employed in a modern Media Technology curriculum:

1 personas and mood boards
2 a design concept and view (generally, in text)
3 task lists, task descriptions and task analysis models
4 usage scenarios and storyboards
5 use cases, activity diagrams, entity relationship diagrams
6 interface sketches or paper prototypes, wireframes and screen designs
7 prototypes, demonstrators and the actual working system

Several of these design representations are exemplified in Figs. 6, 7, 8, 9, 10, 11 and 12, some in abbreviated form, taken from the Media Technology curriculum, in particular from student work and some from research.

Matthew Johnson is a 53 year old college educated male, married for over 20 years with Mary, his childhood girlfriend. They has 2 kids, Bill and Evelyn, ages 15 and 17. Matthew and Mary each have a drivers license but, as highly principled members of the green party, they don't own a car. Matthew rides the metro almost daily one the same journey between home in Leyden, where they live in an monumental house at the canal and work at the city council of Delft. He uses his commuting time to read a newspaper and if wifi is available to check for high-priority email.

Fig. 6. A example description of a persona, a typical and fictive user to represent an important class of user or customers for designers to empathise with.

You don't need a key for the building that you might want to enter: You could use a virtual key to open the door. This virtual key exists in your mobile app. You open the app, click on the door that you want to open. Of course, first the administrator will have to provide you with the proper rights. When you have the proper access rights, the door will open; otherwise it remains closed.

The system is intended to make building-access self-sufficient without a need for a doorkeeper or a reception desk. Furthermore, the administrator always knows who is inside the building.

Fig. 7. A design concept. This design concept exemplifies using an RFID/NFC wireless identification card as a key to a door-lock.

Name	Login as a Student
Summary	the student logs in on the network
Actors	Student
Assumptions	Actor is not yet logged in
Description	Actor enters username and password in the entry-fields and hits "enter" or clicks the button "login"
Exceptions	Wrong username or password
Result	Actor is logged in

Fig. 8. A description of a user task as a use case, a user task description from the point of view of the computer system interacting with the outside world.

Ms. Brown is a vital 72 year old. Two years ago she was diagnosed with type II diabetes for which a diet and medication were prescribed.
On three consecutive days, Ms. Brown's blood glucose level has been slightly higher than normal and today it is rather high. A little alarmed, Ms. Brown presses the help-button on the diabetes assistant and a friendly voice assures her that there is nothing to worry about.
The assistant suggests her to redo the measurement using the little finger of her other hand. Ms. Brown now learns that her blood glucose level is only slightly higher than normal and her assistant asks her to take her pills, including an extra TZD "you know, the big blue one" just to be on the safe side.

Fig. 9. An example user scenario presenting how a user may interact with a webpage or an application to analyse, explain or specify the design product.

The design process of media products is based on prototypes, from low-fidelity prototypes including paper prototypes, mock-ups and sketches to increasingly higher fidelity prototypes including clickable prototypes and the design product itself. Secondly, the media design process is a features-driven process, where each design cycle

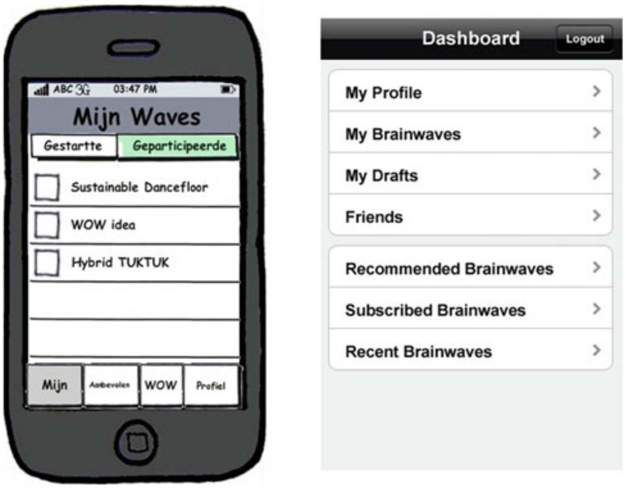

Fig. 10. A paper prototype or a user interface sketch (left) to present test-users or other designers what a design will look like. On the right a clickable prototype to present the interaction design or user-system dialogue of a smartphone application.

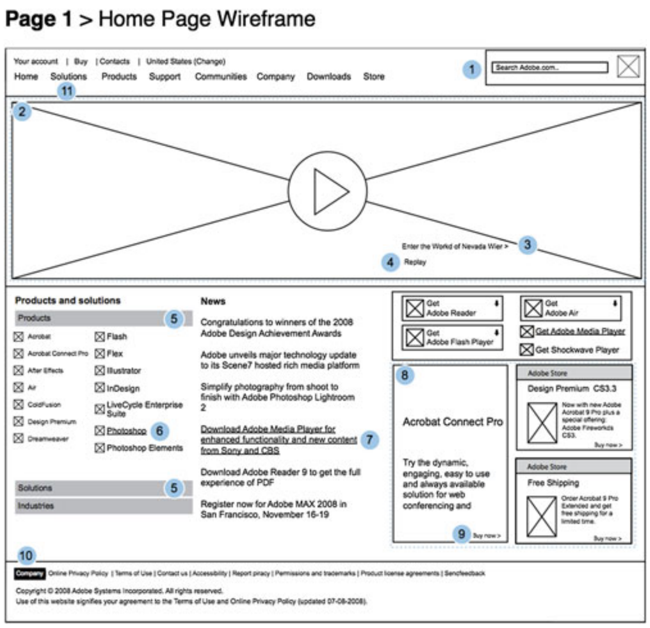
Page 1 > Home Page Wireframe
Your account | Buy | Contacts | United States (Change)
Home Solutions Products Support Communities Company Downloads Store
Search Adobe.com...
Enter the World of Nevada Wier >
Replay
Products and solutions
Products
Acrobat
Acrobat Connect Pro
After Effects
Air
ColdFusion
Design Premium
Dreamweaver
Flash
Flex
Illustrator
InDesign
LiveCycle Enterprise Suite
Photoshop
Photoshop Elements
Solutions
Industries
News
Congratulations to winners of the 2008 Adobe Design Achievement Awards
Adobe unveils major technology update to its Scene7 hosted rich media platform
Simplify photography from shoot to finish with Adobe Photoshop Lightroom 2
Download Adobe Media Player for enhanced functionality and new content from Sony and CBS
Download Adobe Reader 9 to get the full experience of PDF
Register now for Adobe MAX 2008 in San Francisco, November 16-19
Get Adobe Reader
Get Adobe Flash Player
Get Adobe Air
Get Adobe Media Player
Get Shockwave Player
Acrobat Connect Pro
Try the dynamic, engaging, easy to use and always available solution for web conferencing and
Buy now >
Adobe Store
Design Premium CS3.3
Free Shipping
Company
Copyright © 2008 Adobe Systems Incorporated. All rights reserved.
1
2
3
4
5
6
7
8
9
10
11

Fig. 11. A wireframe of a web page presenting the layout of a set of standard web-pages without displaying the actual content.

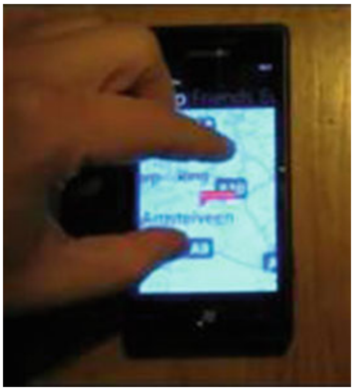

Fig. 12. A demonstrator (or the actual design) as a design prototype which demonstrates actual usage of a product by behaving (almost) as if it is the finished product.

or Scrum sprint focuses on the next most important features to implement. Finally, the media design process is an incremental design process with iteration both during the design process as such, as well as iteration after the design process as such, since maintenance is regarded as including further adaptation to evolving user wishes and tastes.

Media products tend to allow for much flexibility because of the distinction between the ‘front-end’, the website or user interface of the system and the ‘back-end’, the database(s) and the Content Management Systems (CMS) which act as the user interface of the application programmer.

The strict separation of the user interface and the data processing part of the application allows for easy adaptation of the front-end whilst keeping the backend stable. While a website is up and running, it is possible to present different groups of users with a different front-end, depending for example, on the basis of the local webserver they use. Next, data collected online about user preferences, conversion rate or sales figures may be used to choose the most successful front-end design. Naturally, such a process of online optimization is not restricted to a single usability evaluation trial but may take the form of a continuous process of adapting the looks and behaviour of a website or mobile app to the behaviour of its users.

Broos et al. [3] noted that a particular user interface characteristic, such as consistency, is seen as a positive characteristic of user interfaces according to HCI theory but that in designing mobile applications other requirements may become much more important. For instance, in designing a mobile social app for the skating community (board skaters, inline skaters and bmx bicyclists) it is natural to make a distinction between the tasks that users execute while they are actually mobile (hence: actively skating) and those tasks executed when the user is able to pick a steady seat to interact with the application. In the former (mobile) case, interaction should be above all automatic and minimal, utilizing sensor information like GPS location recognition instead of the demanding the user to indicate his or her choices on a keyboard or touch screen [3].

Comparable circumstances evolve in ubiquitous computing and interactive environments, and in intelligent and adaptive interfaces. According to Neerincx et al. [10] task performance should be supported by agents to the extent that human operators have sufficient cognitive capacity to focus on the task at hand; in emergency situations like in marine combat situations, agents should take over all but the most essential tasks in order to optimize total task output.

5 Comparing Design Approaches

When comparing the two design approaches listed, one heavyweight and resting on a design notation as the core of the design process, and one lightweight and utilizing whatever tools seem most appropriate to the design cycle at hand, it will be clear that the media design approach is much more flexible and less regulated and thereby better able to rapidly service any changes in customer wishes and needs, exactly as Schwaber and Beedle [13] tried to address with Scrum.

In designing web applications the media design approach works fine. However, with the ongoing transition towards mobile computing, sentient interfaces and ubiquitous computing, it is our opinion that the iterative features-driven design process has to be further adapted to the new design ecology.

First, increasing focus on user requirements and wishes has increased the employment of co-design and co-creation practices. As a consequence, large parts of application design still take place behind the software engineer's work station and perhaps in the usability lab but increasingly often design 'happens' within the actual context of use.

Secondly, computer applications increasingly make use of sensors in the computer device, in the environment or in both. Consider, for instance, using a GPS service or an application which employs the user's movement patterns, or the simple idea to shut down your phone by putting it on its belly. Of course, the experience that the research field has gathered about such interactive environments is rather limited and, as such, it underlines the need to integrate design with investigating usage and usability aspects in the real world. This provides another argument to remove the distance between the design and the application contexts.

On the basis of the utility of complex formal tools like ETAG in our work on media design, we do not opt for the introduction of new and complex tools to visualise or automate aspects of our design activities; rather, we opt to move design more into the direction of the actual context of use and away from the workstation [8]. Design is about products that enable people to act and interact in the real world, and our design specifications, models and software are just there to make designing such products possible but they are not the essence of what design is all about. To 'situate' design in the context of us, we propose two developments to support this transition.

First, we have recently introduced a sensor lab as a middle-ground between the usability lab and the real world. The sensor lab provides all the facilities for the first crude design iterations, including a range of pre-installed networked sensors and interactive display screens, observation cameras and microphones. In this manner we are able to experiment with and investigate the use of sensors in an environment which also provides usability lab facilities. Next to the ‘sensorlab’, we introduced a ‘fablab’ to extend the design facilities towards interactive objects in general rather then smart phones and other pre-designed interactive objects, and a ‘citylab’ was set up as a place to collect and utilise all kinds of data from the public environment; open data. All these lab facilities enable designers to experiment and do ‘rapid prototyping’ in each design stage, be it conceptualising, functional design, interaction design or tangible design.

Secondly, we investigate the use of self-configuring sensor networks like Almende’s ‘sense-os’ [1]. Networks like these make it possible to hook up one’s mobile phone to a network and to collect on-line sensor and usage data from the phone or other networked sensor devices thus enabling a so-called living lab which acts as a usability lab within the everyday real world environment. Actually, self-configurating networks is just another example of the transition to move our design tools to a next higher level of abstraction: what began with programming by wire and evolved alongside assembly languages and user interface toolkits will certainly move towards self-configuring sensor systems, data resources and user adaptation in the internet of things.

References

1. Almende. Observation Systems. http://www.sense-os.nl/ (2011)
2. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading (1999)
3. Broos, M., van Gammeren, P., van Steenoven, T., de Haan, G.: Creating a context-aware mobile application to enlarge social cohesion: skating together. Accepted for ECCE 2011 - Designing Collaborative Activities, Rostock, Germany, 24−26 August 2011 (2011)
4. Constantine, L.L., Lockwood, L.A.D.: Software for use: a practical guide to the models and methods of usage-centered design. ACM Press, New York (1999)
5. Foley, J.D., Sukaviriya, P.: History, results, and bibliography of the User Inter-face Design Environment (UIDE), an early model-based system for user interface design and implementation. In: Paternó, F. (ed.) Interactive Systems: Design, Specification, and Verification ‘94, pp. 3–13. Springer, Heidelberg (1995)
6. de Haan, G.: ETAG-based design: user interface design as mental model specification. In: Palanque, P., Benyon, D. (eds.) Critical Issues in User Interface Systems Engineering, pp. 81–92. Springer, London (1996)
7. de Haan, G.: DevThis: HCI education beyond usability evaluation. In: Lenior, D., Sturm, J., Mulder, I. (eds.) Proceedings Chi Sparks, Arnhem, The Netherlands, 23 June 2011 (2011)
8. de Haan, G., Choenni, S., Mulder, I., Kalidien, S., van Waart, P.: Bringing the research lab into everyday life: exploiting sensitive environments to acquire data for social research. In: Hesse-Biber, S.N. (ed.) The Handbook of Emergent Technologies in Social Research (Chapter 23), pp. 522–541. Oxford University Press, New York (2011)
9. Moran, T.P.: The command language grammar: a representation for the user-interface of interactive systems. Int. J. Man Mach. Stud. **15**(1), 3–50 (1981)
10. Neerincx, M.A., Lindenberg, J., Grootjen, M.: Accessibility on the job: cognitive capacity driven personalization. In: Proceedings of HCI International 2005, Las Vegas, USA, 22–27 July 2005 (2005)

11. Paternò, F.: Model-Based Design and Evaluation of Interactive Application. Springer, Heidelberg (1999)
12. Payne, S.J., Green, T.R.G.: Task-action grammars: a model of the mental representation of task languages. Hum Comput. Interact. **2**(2), 93–133 (1986)
13. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall, Upper Saddle River (2002)
14. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
15. Tauber, M.J.: ETAG: Extended Task Action Grammar: a language for the description of the user's task language. In: Proceedings Interact' 90, pp. 163–168. North-Holland (1990)