Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3217-3235
submitted: 23/7/08, accepted: 29/10/08, appeared: 1/11/08 © J.UCS

Model-Based and Prototyping-Driven User Interface Specification to Support Collaboration and Creativity

Thomas Memmel
(Workgroup Human-Computer Interaction, University of Konstanz, Germany
memmel@acm.org)

Harald Reiterer
(Workgroup Human-Computer Interaction, University of Konstanz, Germany
Reiterer@inf.uni-konstanz.de)

Abstract: When the user interface is specified, a picture is worth a thousand words, and the worst thing one can do is write a natural-language specification for it. Because this practice is still common, it is a challenging task to move from text-based requirements and problem-space concepts to a final UI design, and then back again. However, this activity is required frequently and is necessary to drive creative ideas. In our research we found that advanced UI specifications should therefore be made up of interconnected artefacts that have distinct levels of abstraction. With regards to the transparency and traceability of the rationale of the specification process, transitions and dependencies must be visual and traversable. For this purpose, we introduce a model-based user interface specification method and a corresponding experimental tool that interactively integrates interdisciplinary and informal models with different levels of fidelity of user-interface prototyping. With innovative styles of interaction and user input, our proposed tool supports the collaboration required in a multidisciplinary context.

Keywords: model-based development, user interface design, prototyping, specification
Categories: H.5.2, H.5.3

1 Introduction

It is generally recognized by both software practitioners and Human-Computer Interaction (HCI) specialists that structured approaches are required to model, specify, and build interactive systems with high usability [Metzker et al., 2002]. This structure should be reflected in the Software Development Life Cycle (SDLC). Nevertheless, in many organizations, UI design is still an accidental or opportunistic by-product and HCI methods are not sufficiently embedded in the overall SDLC. If they are integrated, their contribution remains marginal, thus reducing the expected positive impact on software quality. This reality can be explained by the fact that most Integrated Development Environments (IDEs) are inappropriate for supporting actors from different disciplines in designing interactive systems. Formal UI tools prevent many actors from taking part in collaborative design if they do not have adequate knowledge of specific terminologies. On the other hand, being too informal leads to misunderstandings and conflicts in communication with programmers. Moreover, on further examination, many tools turn out to be more focused on requirements management than on providing support in extracting requirements from user needs and translating them into good UI design. After all, despite - or perhaps precisely because of - the vast functionality of many tools, the outcome is often unsatisfactory in terms of UI design, usability and aesthetics. This is described as the high threshold - low ceiling phenomenon of UI tools [Campos et al., 2004].

1.1 Actors in the UI specification process

Over the last 3 years, we observed UI development practice in the German automotive industry [Memmel et al., 2007a; Memmel et al., 2007c]. As a consequence of the lack of appropriate tools, many actors tend to use tools they are familiar with and which can be categorized as being low threshold – low ceiling – narrow walls IDEs, a phenomenon that has been thoroughly observed by [Campos et al., 2004]. We distinguish between two different populations of tool-users, which can be assigned to two different areas of corporate UI development projects: (1) Client: business personnel, marketers, domain experts, or HCI experts use Office-like applications such as MS Word or MS PowerPoint [Memmel et al., 2007a] to document user needs and context of use in order to define the problem-space. They will translate the needs as analyzed, and their contextual conditions, into general usage requirements and evaluate their work at several quality gates. At this stage, responsibility is typically shared with, or completely passed on to, an IT supplier. (2) Supplier: actors with a sophisticated IT background (e.g. programmers or designers) translate usage requirements into UI and system requirements, deliver prototypes and finalise the outcome in a UI specification. Working with UI builders, and using more formal, precise and standardized notations, they narrow the solution space towards the final UI. Ultimately, the described assignment of responsibility leads to a project environment that is propelled by specification-driven UI prototyping and UI development. This means that it is primarily the specification that drives the subsequent prototyping process. In specification-driven prototyping cultures, end users (i.e. the client) can first access prototypes of the software system only after the specification sheet has been consolidated and the supplier has started working [Schrage, 1999].

1.2 Shortcomings of current UI specification practice

The difference between these groups of actors tends to result in a mixture of formats. This makes it difficult to promote concepts and creative thinking down the supply chain without media disruptions and loss of precision [Memmel et al., 2007a]. The following negative factors therefore contribute to UI development failure: (1) The lack of a common course of action and the use of inappropriate, incompatible terminologies and modelling languages [Zave et al., 1997] that prevent even the minimum levels of transparency, traceability and requirements-visualization that would be adequate for the problem. (2) The difficulty in switching between abstract and detailed models due to a lack of interconnectivity [Campos et al., 2006]. (3) The difficulty of travelling from problem space to solution space, a difficulty that turns the overall UI development into a black-box process. (4) The burial of mission-critical information in documents that are difficult to research and have very awkward traceability. Experts are overruled when the UI design rationale is not universally available in the corresponding prototypes. (5) The perpetuation of unrecognized cross-purposes in client and supplier communication, which can lead to a premature change or reversal of UI design decisions, the implications of which will not be realized until later stages. (6) The resulting misconceptions that lead to costly change requests and iterations, which torpedo budgets and timeframes, and endanger project goals.

Because of the immaturity of their UI development processes, industrial clients determined on a shift of responsibility. In our research for Dr. Ing. h. c. F. Porsche AG and Daimler AG, we found the following sticking points that tend to change current UI specification practice. (1) Due to the strategic impact of many software products, clients want to increase their UI-related competency in order to reflect corporate values by high UI quality [Memmel et al., 2007c]. (2) Whereas conceptual modelling, prototyping or evaluation have always been undertaken by suppliers, the client himself now wants to work in the solution space and therefore needs to develop the UI specification in-house [Memmel et al., 2007a]. This induces a prototyping-driven specification culture [Schrage, 1999]. (3) The role of the supplier becomes limited to programming the final system. The client can identify a timetable advantage from this change, and an important gain in flexibility in choosing her suppliers. Having an in-house competency in UI-related topics, the client becomes more independent and can avoid costly and time-consuming iterations with external suppliers. (4) It is nearly impossible to specify a UI with Office-like applications. The existing actors, who are nevertheless accustomed to text-based artefacts, now require new approaches. The task of learning the required modelling languages and understanding how to apply these new tools must not be an unreasonably difficult one.

1.3 Tool support that is adequate for the problem

This cultural change must be supported by an integrating UI tool that allows the translation of needs into requirements and subsequently into good UI design. In Table 1 we present a condensed overview of relevant UI tool requirements.

In this paper we present both a set of models and a corresponding tool named INSPECTOR, still under development, which are designed to support interdisciplinary teams in collaboratively gathering user needs, translating them into UI-related requirements, designing prototypes of different fidelity and linking the resulting artefacts (i.e. a combination of expression and medium [Brown et al., 2008]) to an interactive UI specification. The term interactive refers to the concept of making the process visually externalized to the greatest extent possible. This concerns both the artefacts and the medium of the UI specification itself. The latter should no longer be a text-based document, but a running simulation of how the UI should look and feel. Accordingly, we extend the meaning of UI prototypes to also include the provision of access to information items below the UI presentation layer. Being interactively connected, all of the ingredients result in a compilation of information items that together drive creativity and are necessary to specify the UI (Table 2). In Section 2 we link our research to related work. Section 3 presents the common denominator in modelling that we developed. We explain how our tool, called INSPECTOR, will utilize the resulting interconnected hierarchy of notations and UI designs. In Section 4 we present the results of two evaluation studies. Accordingly, in Section 5 we deduce some aspects of our future work. The article ends with a summary and conclusion.

Purpose/Added Value	**Tool Requirement**
Traceability of design rationale; transparency of translation of models into UI design	Switching back and forth between different (levels of) models
Smooth transition from problem-space concepts to solution space	Smooth progression between abstract and detailed representations visualizes the model-based fashion of the process
HCI experts can build abstract and detailed prototypes rapidly for continuous UI simulation	Designing different versions of a UI is easy and quick, as is making changes to it, and thereby supports creativity
Provide support for design assistance and creative thinking for everybody; all kinds of actors can proactively take part in the UI specification	Concentration on a specific subset of modelling artefacts, which can be a UML-like notation or one that best leverages collaboration
The early detection of usability issues prevents costly late-cycle changes	Allowing an up-front usability evaluation of the UI; providing feedback easily

Table 1: Requirements for UI tools for interactive UI specification; on the basis of [Memmel et al., 2007a; Nunes et al., 2004; Campos et al., 2006]

Interactive UI Prototypes	**Interactive UI Specifications**
Vehicle for requirements analysis	Vehicle for requirements specification
Exclusively models the UI layer; may be inconsistent with specification documents	Allows drill-down from UI to models; relates UI to requirements and vice versa
Either low-fidelity or high-fidelity	Abstract first, specification design later
Supplements text-based specification; mostly **driven by specification**	Widely substitutes text-based specification; **driven by prototyping**
Design rationale saved in other documents	Incorporates design knowledge and rationale

Table 2: Main differences between prototypes and interactive UI specifications

2 Related Work

Campos and Nunes presented the tools CanonSketch and TaskSketch [Campos et al., 2006]. CanonSketch was the first tool that used canonical abstract prototypes and an UML-like notation, supplemented by a functioning HTML UI design layer. TaskSketch is a modelling tool that focuses on linking and tracing use cases, by means of which it significantly facilitates development tasks with an essential use-case notation. Altogether, TaskSketch provides three synchronized views: the participatory view uses a post-it notation to support communication with end-user and clients, the task-case view is targeted towards designers and is a digital version of index cards (well-known artefacts of usage-centred or agile developers) and the UML activity diagram view is adequate for software engineers. As we will see in this paper, we closely concur with the concepts of these tools, but our approach differs in some important areas. Firstly, and in contrast to CanonSketch, we also support detailed UI prototyping because we found that the high-fidelity externalization of design vision is especially important in corporate UI design processes. Secondly, we provide more ways of modelling. INSPECTOR integrates earlier text-based artefacts, as well as task models and interaction diagrams. Some of them are also grounded in usage-centred design, but we focused on agile models as they proved to be helpful in bridging the gaps between the disciplines (see Section 3).

The tools DAMASK [Lin et al., 2002] and DENIM [Newman et al., 2003] use a Zoomable User Interface (ZUI) approach for switching between different levels of detail through a visual drill-down process. Based on our own experience with ZUIs, we followed a consistent implementation of this technique. Calvary et al. [Calvary et al., 2003] presented the CAMELEON reference framework, which proposes four levels of abstraction for UI tools: tasks and concepts, abstract UI design, detailed UI design, and the final UI. We will show that INSPECTOR supports this framework very well by the nature of the layers of abstraction used and the ZUI approach applied. However, as INSPECTOR is focused on UI specification rather than on actual UI development, it supports the final UI stage by means of UIs to other tools in the supply chain. With respect to DAMASK and DENIM, INSPECTOR borrowed the idea of using animations to support transitions between contexts of use: when an actor needs to switch from one view to another, INSPECTOR applies a zoom-in, zoom-out technique so as to preserve continuity between the contexts of use, which has been largely demonstrated as a positive impact in SDLC [Lin et al., 2002].

3 A Common Denominator In UI-related Modelling

An advanced IDE must be able to support all actors in actively participating in the UI specification process (Table 1). This requires it to deploy modelling techniques that can be used easily by everybody. We know that the Unified Modelling Language (UML) is a weak means of modelling the UIs of interactive systems [Sutcliffe, 2005]. As well as its shortcomings in describing user interactions with the UI, its notation also overwhelms most actors with too much (and mostly unnecessary) detail [Ambler, 2004]. Designing UIs is an interdisciplinary assignment and many actors might be left behind due to unnecessary formality. Altogether, both UML and Office-like artefacts are inadequate means for the specification of the look and feel of interactive UIs. Conversely, in agile development useful lightweight artefacts well support collaboration and design [Brown et al., 2008].

3.1 Bridging the gaps with Agile Modelling

The identification of adequate means of modelling for UI specifications is very much related to the ongoing discussion on bridging the gaps between HCI and SE. This discussion is also propelled by the very difference in the way experts from both fields prefer to express themselves in terms of formality and visual externalization. HCI and

SE are recognized as professions made up of very distinct populations. In the context of corporate UI specification processes as outlined in Section 1, modelling the UI also requires the integration of the discipline of business-process modelling (BPM). With regards to UI development, business processes can play an important role. It is important to maintain a close connection between business processes and the UI. Business analysts analyze and model business processes that serve as requirements for the UI. Having high UI quality is normally consistent with business needs. Product knowledge must not be owned just by business analysts, but must be shared with other stakeholders. Concerning the UI, interaction designers must have a clear understanding of the processes of an enterprise. The information buried in various artefacts of BPM must be externalized visually to be able to design for user experience. Sousa, Mendonca et al. [Sousa et al., 2008] propose task models as an intermediate modelling language that could bridge process modelling and UI development.

The interaction layer - as interface between system and user - is the area where HCI, SE and BPM are required to collaborate in order to produce high-quality UIs. As actors in corporate UI specification processes come from all three disciplines, the question is which modelling notations are adequate to extend and align their vocabulary. As we found in our previous research, agile methods are close to HCI practice [Memmel et al., 2007b]. [Brown et al., 2008] found that agile artefacts, for example use cases [Ambler, 2002; Ambler, 2004] and scenarios [Barbosa et al., 2003, Rosson et al., 2002] successfully drive discussions between software developers and designers. Personas [Beyer et al., 1998] additionally raise empathy for user needs and identification with user roles. Agile development has always emphasized collaboration and therefore represents a promising pathfinder for a course of action common to all three disciplines.

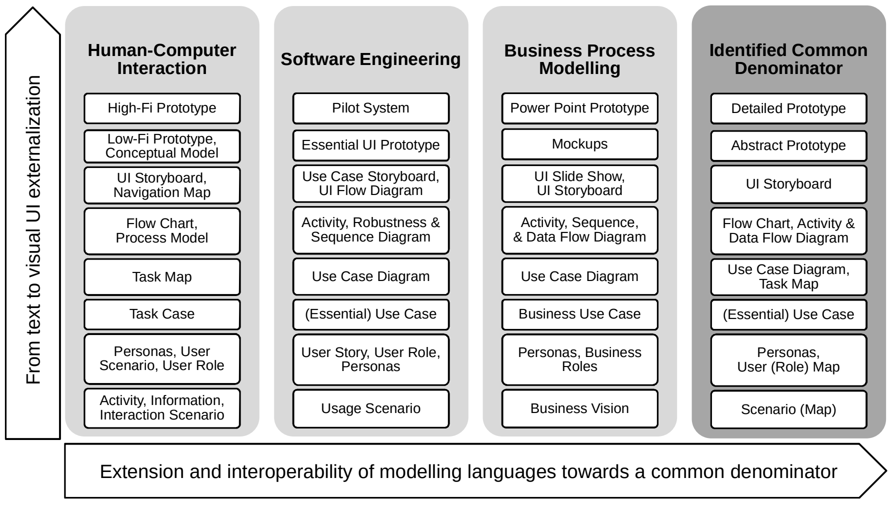
From text to visual UI externalization
Human-Computer Interaction
High-Fi Prototype
Low-Fi Prototype, Conceptual Model
UI Storyboard, Navigation Map
Flow Chart, Process Model
Task Map
Task Case
Personas, User Scenario, User Role
Activity, Information, Interaction Scenario
Software Engineering
Pilot System
Essential UI Prototype
Use Case Storyboard, UI Flow Diagram
Activity, Robustness & Sequence Diagram
Use Case Diagram
(Essential) Use Case
User Story, User Role, Personas
Usage Scenario
Business Process Modelling
Power Point Prototype
Mockups
UI Slide Show, UI Storyboard
Activity, Sequence, & Data Flow Diagram
Use Case Diagram
Business Use Case
Personas, Business Roles
Business Vision
Identified Common Denominator
Detailed Prototype
Abstract Prototype
UI Storyboard
Flow Chart, Activity & Data Flow Diagram
Use Case Diagram, Task Map
(Essential) Use Case
Personas, User (Role) Map
Scenario (Map)
Extension and interoperability of modelling languages towards a common denominator

Figure 1: Towards a common denominator in interdisciplinary UI-related modelling

Holt [Holt, 2005] presents a BPM approach that is based on UML class, activity, sequence and use-case notations. Ambler based his agile version of the Rational Unified Process (RUP) on a similar, but less formal, BPM approach [Ambler, 2002]. In general, agile approaches already exist in HCI [Memmel et al., 2007b], BPM [Ambler, 2002] and SE [Beck, 1999] and we can define a common denominator for all three disciplines. We keep this denominator as small as possible. We filter out models that are too difficult to be understood by every actor. We do not consider models that are more commonly used to support actual implementation or that have been identified as mostly unnecessary by Agile Modelling [Ambler, 2004, Ambler, 2002]. IT suppliers can deduce the structure of the UI much better from the resulting interactive specification than they can from Office-like documents.

We integrate different levels of modelling abstraction to visualize the flow from initial abstract artefacts to detailed prototypes of the interaction layer. On the vertical axis in Fig. 1 we distinguish the models according to their level of abstraction. Models at the bottom are more abstract (i.e. text-based, pictorial), whereas those at upper levels become more detailed with regard to the specification of the UI. On the horizontal axis, we identify appropriate models for UI specification. Accordingly, we differentiate between the grade of formality of the models and their purpose and expressivity. The models with a comparable right to exist are arranged at the same level. At each stage we identify a common denominator for all three disciplines as a part of the evolving interactive UI specification.

3.2 Text-based notations of needs and requirements: personas and scenarios

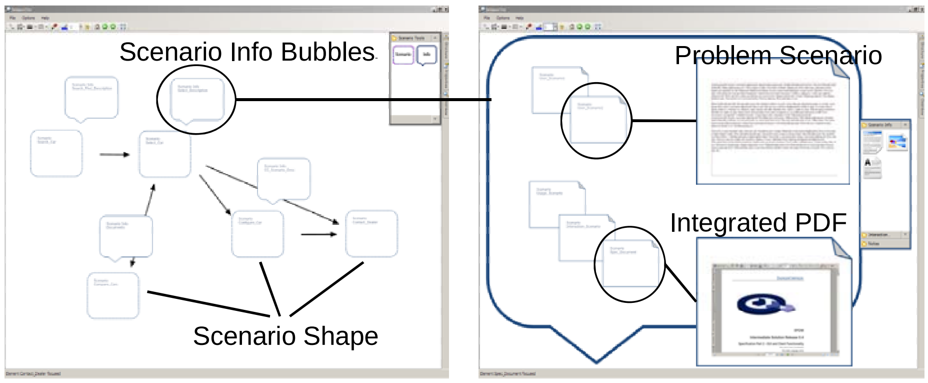

*Figure 2**:** Scenario map as entry stage to the modelling process (left); scenario info-bubble (right)*

Text-based notations can be used at any stage to document early usability attributes (usability and user experience goals, constraints, etc) with INSPECTOR's information bubbles (Fig. 2, left). For describing users and their needs, HCI recognizes user profiles, (user) scenarios [Rosson et al., 2002], role models [Constantine et al., 1999], and personas [Beyer et al., 1998]. Roles and personas are also known in SE and BPM and are therefore appropriate for initial user-needs modelling. As an interdisciplinary modelling language, research suggests scenarios

[Barbosa et al., 2003] - known as user stories (light-weight scenarios) in agile development [Beck, 1999]. In SE, scenarios – as a sequence of events triggered by the user – are generally used for requirements gathering and for model checking. Such a scenario is used to identify a thread of usage for the system to be constructed and to provide a description of how the system will be used. HCI applies scenarios to describe in detail the software context, users, user roles, activities (i.e. tasks), and interaction for a certain use-case. BE uses scenario-like narrations to describe a business vision, i.e. a guess about users (customers), their activities and interests. Altogether, written stories are for raising problems, defining scope, articulating early requirements, and keep the design activity interesting and fun [Brown et al., 2008], which in turns drives creativity. On starting INSPECTOR, the user can create a scenario map that relates all scenarios that will be modelled (Fig. 2, left). The user can first describe a single scenario in a bubble shape (Fig. 2, right). For this purpose, INSPECTOR provides a built-in text editor with appropriate templates and enables the direct integration of existing requirement documents into its repository. Later, the user will zoom-in and fill the scenario shape with graphical notations and UI design (see Section 3.3).

3.3 Graphical notations: requirements, usage and behaviour modelling

Entering this stage, the user needs artefacts that support the important process of translating needs into requirements. Role maps [Constantine et al., 1999] help to relate created personas to each other (Fig 3, left). Although different in name, task cases (HCI), essential-use cases (SE), and business-use cases (BPM) can be created in a classical use-case notation (Fig. 3, centre).

Use-case diagrams (SE, BE) overlap with use-case and task maps (HCI) [Constantine et al., 1999]. The latter also help to separate more general cases from more specialized (essential) sub-cases. We considered different models for task and process modelling and, following Ambler [Ambler, 2002; Ambler, 2004], we again selected related modelling languages.

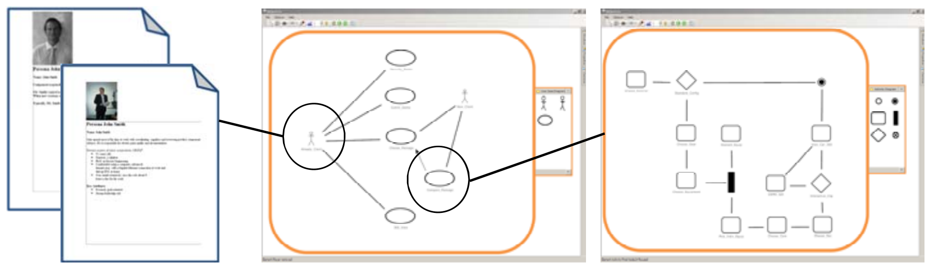

Figure 3: Personas (left), Use-Case Diagram (center); Activity Diagram (right)

Activity diagrams (Fig. 3, right) are typically used for business-process modelling, for modelling the logic captured by a single use-case or usage scenario, or for modelling the detailed logic of a business rule. They are the object-oriented equivalent of flow charts and data-flow diagrams.

Data-flow diagrams model the flow of data through the interactive system. With a data-flow diagram, actors can visualize how the UI will operate depending on external entities. For the storyboard layer we decided to keep the typical UI storyboards we know from HCI [Beyer et al., 1998]. The storyboard serves as interface layer between needs and requirement models and the UI design (Fig. 4).

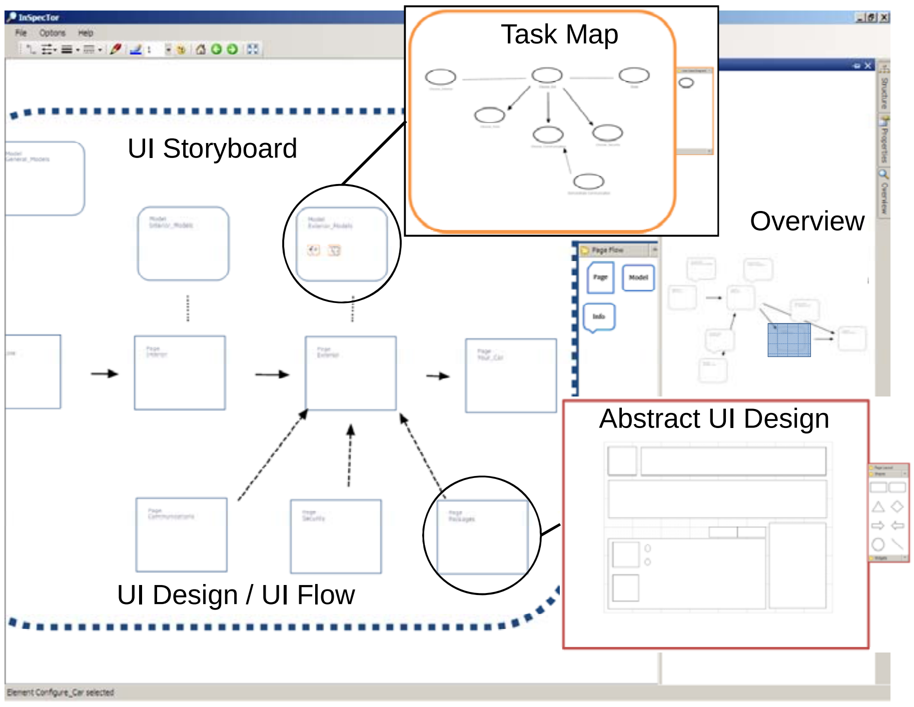

Figure 4: UI storyboard with UI design and models; magnified areas show embedded artefacts

3.4 UI prototyping and simulation: modelling look and feel

Prototypes are already established as a bridging technique for HCI and SE [Zave et al., 1997; Blomkvist, 2005]. HCI mainly recognizes them as an artefact for iterative UI design. Avoiding risk when making decisions that are difficult to retract is a reason why prototyping is also important for business people. Accordingly, we chose prototypes as a vehicle for abstract UI modelling. Sketches are required to make note of good design ideas [Brown et al., 2008]. Low-fidelity prototypes will help to design and evaluate the UI at early stages and they support traceability from models to design. Alternate designs can be maintained in the specification landscape to safeguard the design rationale. UI elements can be assembled to templates in order to ease and speed up the design process. The visually most expressive level is the high-fidelity UI prototyping layer (Fig. 5). It serves as the executable, interactive part of the UI specification and makes the package complete.

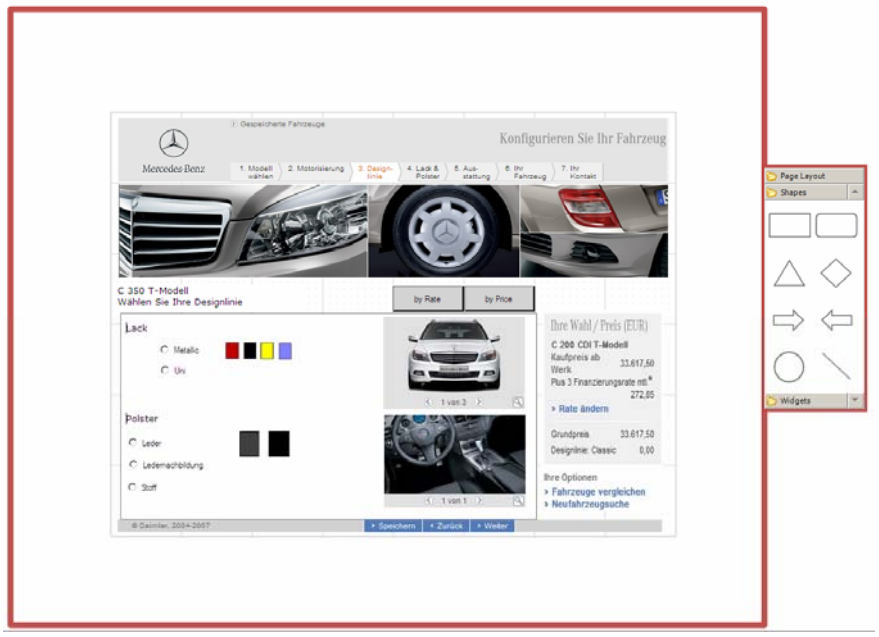

Figure 5: INSPECTOR-made high-fidelity UI design

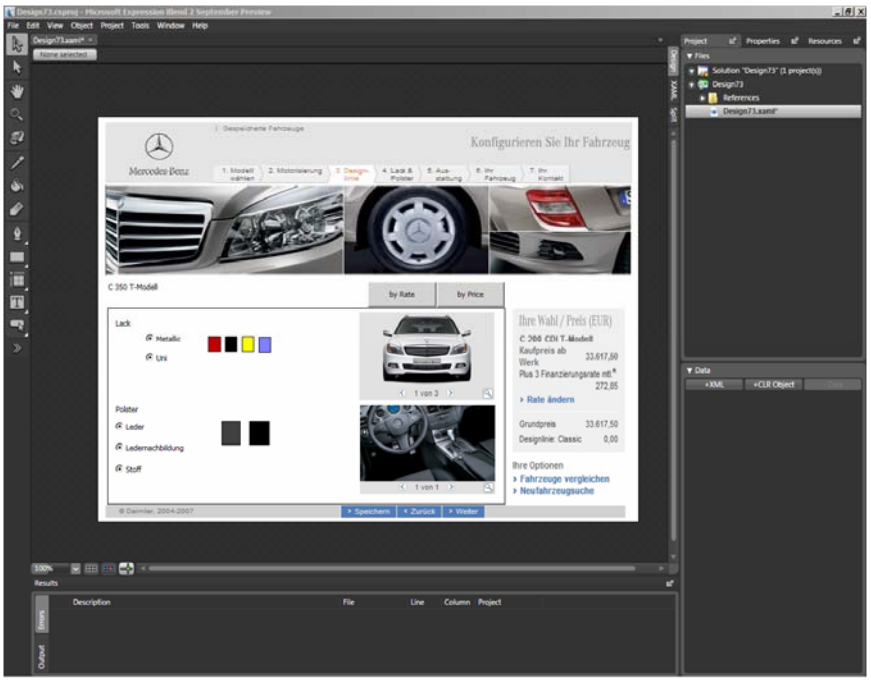

Figure 6: INSPECTOR-made UI design opened in Microsoft Expression Blend

From this point, actors can then explore, create and change models by drilling down to the relevant area of the UI specification. Moreover, programmers can pop-up the interactive UI specification to get guidance on the required UI properties. Therefore, all UI designs that have been created can be saved in two XML formats. On the one hand, the XAML export guarantees the reusability of the specified UIs during the development by the supplier. The XAML code can, for example, be imported to MS Expression Blend (Fig. 6). The XAML helps to provide simulations of the UI in a web browser such as Microsoft Internet Explorer. On the other hand, as a member of the UsiXML supply chain [Memmel et al., 2008], INSPECTOR can contribute to the early phases of needs analysis and requirements engineering. With its UI design layer, INSPECTOR can also be compared to tools such as GrafiXML [Lepreux et al., 2006].

3.5 Travelling through the UI specification process with INSPECTOR

INSPECTOR is based on the metaphor of a whiteboard, which is a very common tool in collaborative design environments. Information can flow from source to need, from abundance to absence, and both ends can influence the flow. The flow is also affected by temporal and spatial arrangements. Designers therefore heavily rely on whiteboards for telling stories and recording their design ideas and design rationale [Brown et al., 2008]. Basically, actors can therefore apply the models and design capabilities of INSPECTOR in arbitrary order along the UI specification process. However, the scenario map is very well suited to work on early assignments of UI specification processes. Usability and user-experience goals, business and design vision as well as reusable requirements can be captured within the information bubbles at the scenario layer. At this initial stage, problem scenarios can be textually documented. They will be enriched by concrete artefacts at the UI storyboard layer, which functions as the mediator between interconnected models and designs. It encapsulates the collection of linked and interrelated artefacts by means of panning and zooming as major interaction techniques [Lin et al., 2002; Newman et al., 2003]. This provides actors with a feeling of diving into the information space of the UI specification whiteboard.

The appearance of INSPECTOR’s UI is based on a linear scaling of objects (geometric zooming) and on displaying information in a way that is dependent on the scale of the objects (semantic zooming) [Ware, 2004]. Automatic zooming automatically organizes selected objects on the UI. Animated zooming supports the user in exploring the topology of an information space and in understanding data relationships. For switching between models and UI designs, the user can manually zoom in and out and pan the canvas. During user modelling, for example, a user shape can be linked to, and be part of, user roles, personas, and use-cases. Zooming-in on a user shape reveals more details about the underlying personas. The use-case shapes can be part of a superordinate task map and can be linked accordingly (Fig. 7).

Moreover, zooming in a particular case could link to an essential use-case description and reveal more detail on user and system responsibilities. At this stage, activity and data-flow diagrams help during interaction modelling. The user can link every model to UI designs of different fidelity and vice versa (Fig. 7). During modelling, or while traversing relationships by panning and zooming, hints about the current zoom factor and the current position in the information space can be given in order to avoid disorientation. A common way of supporting the user's cognitive (i.e. spatial) map of the information space is an overview window (Fig. 4). Navigating between artefacts can be an extensive task, however, if objects are widespread in terms of being some distance along the three dimensions of the ZUI canvas. For a much faster navigation, actors can switch between artefacts with a tree-view explorer that allows a jump zoom into areas far removed from the current user focus.

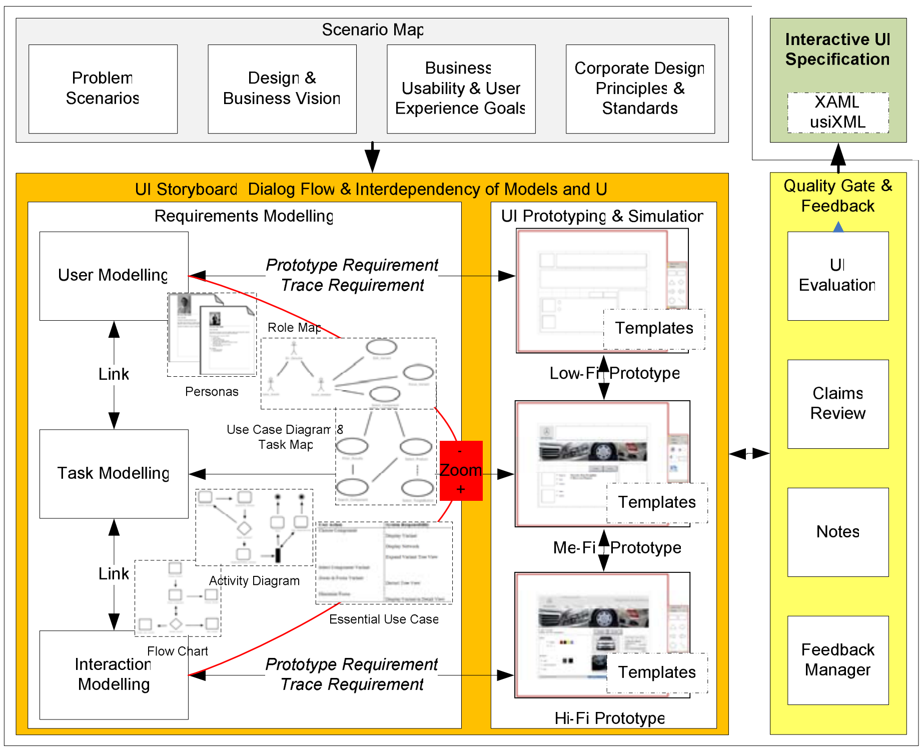

Figure 7: *Exemplified modelling and design throughput with INSPECTOR*

In order to support the assessment of the UI specification quality, we are also working on a feedback component for INSPECTOR. Annotations can be attached to any canvas object. They will be used to review requirements models, to integrate results of UI evaluation studies or to incorporate notes about trade-offs or design decisions. Annotations will be accessible through a management component, which allows a direct zoom-navigation to the artefacts concerned. This well supports the identification and decomposition of contradictions, as they frequently occur during reflection phases [Brown et al., 2008]. Equally important for design rationale, the feedback will also be stored in the UI specifications such as XAML.

4 Expert Feedback and Usability Study

4.1 Expert Interviews

We interviewed software and UI specification experts (n=6) in a questionnaire-based usability study. The participants were introduced to INSPECTOR through a short demonstration, a video and a supplementary text explaining the motivation for our approach. Each expert was provided with an installation of the tool and had two weeks to return their feedback by means of a questionnaire that was divided into 5 parts.

The first part was designed to (1) identify the field of activities of every respondent, (2) get an overview of the models and tools typically applied, and (3) get an assessment of difficulties along the supply chain. The second to fourth parts asked about INSPECTOR in terms of (1) the applicability of the modelling notations, (2) the completeness of the UI design capabilities and their practicability for UI evaluation, and (3) the assessment of the tool's general usability and the user experience provided. The fifth part asked if INSPECTOR could, in general, improve the UI specification practice.

Questionnaire topic	**Avg.**
Ability to integrate documents and logic with INSPECTOR	3.66
Opportunity to capture conceptual and schematic ideas	3.83
Support for user, task and interaction modelling	4.00
Link models and thereby increase the traceability and transparency	3.66
Text-based and graphical requirements modelling (aggregated)	3.79
Accessibility of the prototyping features	3.16
Functionality provided at the UI design layer	3.40
Applicability of the UI designs for usability evaluations	3.33
Possibility to link UI designs in order to create a simulation	3.25
Overall UI prototyping capabilities (aggregated)	3.28
Opportunity to get both overview and detail on the specification space	3.33
Helpfulness of the zoom-interaction style during prototyping and modelling	3.00
Support for switching between created artefacts	3.50
Accessibility of all necessary information on the zoom canvas	3.50
Overall rating of the interaction with INSPECTOR (aggregated)	3.33
Overall contribution of INSPECTOR to existing UI specification practice	3.83
Improvement of work style through a combination of different models with multi-fidelity UI design	4.83

Table 3: Overview on feedback; average points based on a 5-point Likert scale

With regards to the results of our survey (see Table 3), all respondents have stated that INSPECTOR, as a tool that combines models with UI Design, contributes great value to their work style (average 4.83 pts; on a 5-point Likert scale). The added value was particularly identified in terms of an increased coherence of models and design artefacts, whereby INSPECTOR enhances traceability and transparency. Even the very early version of INSPECTOR was therefore already expected to be able to improve existing UI specification practice (average 3.83 pts). The participants of the study were quite satisfied with INSPECTOR's support for text-based and graphical requirements modelling (average 4.00 pts).

Nevertheless, the feedback pointed to the necessity for a better linking functionality between the modelling artefacts. Some experts stated that while creating a UI design, the interaction with INSPECTOR could be enhanced by a contextual layer. This could give the expert the chance to easily cross-check the design with underlying models. Instead of frequently jumping back and forth on the canvas, it is then possible to temporarily visualize models and UI concurrently. Consequently, we implemented a visualization that highlights all outgoing and incoming links of a model in order to enhance traceability. Due to the experimental stage of INSPECTOR’s design and prototyping facilities, the experts missed some important features such as master components and templates. These are needed to allow for rapid prototyping and quick generic changes. In addition to a copy & paste mechanism that was required for the UI design layer, we therefore also implemented support for grouping UI elements and storing them in a template repository. In order to improve the utility of INSPECTOR during usability evaluations of modelling and design artefacts, we also developed an annotation component. During meetings, discussions and feedback sessions, sticky notes can be attached to all artefacts on the specification canvas. This allows the recording of feedback and design decisions for later consideration during subsequent specification tasks. The notes can be accessed in a spreadsheet component that allows sorting and filtering, as well as jump navigation towards them.

4.2 Long-term Diary Study

Other usability issues concerned the general interaction with the tool and were similar to those found during a diary study. As proposed in [Shneiderman et al., 2006] we used diaries to evaluate the long-term usability of INSPECTOR. We therefore used INSPECTOR during an interaction design lecture. Three groups of computer science and HCI students (n=8) were asked to use the tool during a use-case study on the specification of rear-seat entertainment systems.

For a period of three weeks, every student wrote their own diary to give insight into (1) the kind of models created, (2) additional tools that were applied, (3) problems that occurred, (4) ratings of the user experience, (5) general issues and opinions about the tool. We decided on the diary study in order to evaluate INSPECTOR over a longer period of time. Because we were interested in how the empirical results change with the duration and intensity of usage, we preferred a long-term study to classical usability tests. In weekly workshops, we discussed the intermediary results and recorded the issues for subsequent correction.

By means of the diary study, we found, for example, that objects on the ZUI canvas occasionally behaved inconsistently after the tool was used for several hours and an extended amount of zoom operations had been performed. Students also reported issues with integrated external documents (PDF, Word, etc.), when these were repeatedly saved and opened. This led to a disarrangement of the XML structure in saved project files and significantly prevented a fluent and enduring work style. To have identified these problems in a much shorter lab-based usability study would have been pure chance. Thanks to the diary study, we were able to solve these issues quickly.

Moreover, we found that some participants preferred to create the first abstract prototypes initially with paper and pencil. We realized that the use of the built-in sketching mechanism increased as soon as we provided a pen tablet as an input device. In addition, it proved to be very difficult to rapidly prototype UIs with point and click interaction on the canvas. We will therefore evaluate different pen tablet technologies that we could permanently combine with INSPECTOR. This will significantly increase the application performance during design sessions. In addition, students were initially not comfortable with all the notations provided and required assistance on their proper application. We addressed this issue by making a start on including a help feature that explains notations as well as their scope of application. In addition, we enhanced the affordance of templates for personas or essential-use cases, for example, to ease the understanding of the artefacts.

Ultimately, the diary study and the upgrades resulted in an improvement of the feedback on the tool usability. Rated with an average of 1.75pts (std. 0.46) (on a 5-point Likert scale) after the first week and 3pts (std. 0.00) after the second, participants assessed INSPECTOR with an average of 4.25pts (std. 0.46) at the end of the study. A repeated-measure ANOVA revealed a significant main effect for the rating across the weeks ($F(2,14)=105.00$, $p<0.001$). Furthermore the differences between each week are also very significant statistically (week 1 vs. week 2: $F(1,7)=58.33$, $p<0.001$; week 2 vs. week 3: $F(1,7)=58.33$, $p<0.001$). The according inter-rater agreement was assessed by calculating the intraclass correlation. This revealed a significant correlation of 0.99 ($p=0.000$), indicating a high homogeneity in subjects' ratings of the system across the three weeks.

5 Future Work

In meeting or decision-room set-ups, INSPECTOR supports collaboration and decision-making. Users can cooperatively work on requirement models or UI designs during brainstorming sessions. Utilized as an electronic whiteboard, INSPECTOR records all created artefacts in a structured manner. Actors can also work asynchronously using their own workspace, for example on a desktop installation. Modelled artefacts are then exported into XML documents and re-imported into a shared workspace, which resembles the common design rationale. Initial experimental setups with our high-resolution powerwall installation (4640 x 1920 pixels) allowed a comprehensive view on our zoomable specification space (see Fig. 7). This high-resolution display supports our ZUI approach by displaying a wide range of artefacts and relations (overview) to all actors. Laser-pointer interaction enables an easy-to-use cooperative and more collaborative interaction style (see Table 4). Through point and click operations, actors explore and manipulate the UI specification space. In order to ease the collaboration with multi-modal input devices and to align INSPECTOR more consistently with an electronic whiteboard metaphor, we will follow a more straightforward zoom approach. Therefore, we will partly dissolve the nesting of modelling and design objects on the specification canvas in favour of a large zoomable object-information landscape. The handling of complexity will then be possible through search and filter functions, which provide quick access to artefacts.

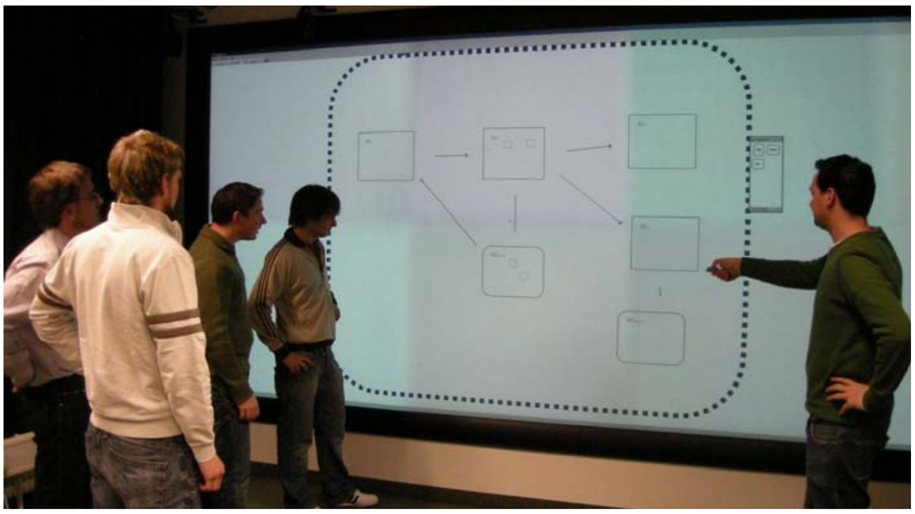

Figure 7: Utilizing INSPECTOR for collaborative meetings at a megapixel powerwall with laser-pointer interaction

Aspect	**Detailed information**
Simulation	Simulations propel the design process. An easy-to-use UI design layer must provide support for prototyping-driven development
Creativity	By interfaces to multi-modal input devices, such as laser-pointer or table-tops, creativity can be well supported. Depending on the situation, suitable input devices help to model what is required
Collaboration	Simulation and support for creativity help to establish a collaborative UI specification style. By using different input devices and large high-resolution displays, actors can work in a computer-aided design room, equipped with an artefact repository. For the latter, we envision a database-supported versioning system that allows management and comparison of the stored artefacts.

Table 4: Overview on future work on INSPECTOR

In order to provide the whole functionality of INSPECTOR at the high-resolution display, multi-modal input devices are necessary to foster creativity and collaboration. In addition to the laser-pointer as an input device, we therefore consider a PDAs or iPhone for text-input (e.g. writing text or annotations) and optionally for panning and zooming as well. Moreover, several stakeholders could work in parallel with INSPECTOR using multiple devices, such as laser-pointer or table-top, at the same time. Naturally, the UI of INSPECTOR must be adapted in some extent to be usable without mouse and keyboard in every situation. Because browsing menu structures is an ungrateful activity using a laser-pointer, interaction with INSPECTOR could as well be supported by speech-recognition. After all, a thoughtful combination of modalities for interacting with INSPECTOR will be a major part of our future research.

6 Summary

Based on our experience in UI specification and design, we have come to the conclusion that the typical methods and tools available are not adequate. A recent study from [Brown et al., 2008] has outlined that the key tools in design meetings are sketches (i.e. prototypes or agile graphical notations), lists (e.g. essential use cases) and design stories (i.e. text-based artefacts). Their role extends beyond representing UI design in terms of noting good ideas and making a point. In this context, whiteboards help to focus attention and serve as collaborative design repository. With INSPECTOR, actors are supported in applying informal models they are familiar with, and are given the opportunity of UI prototyping with different fidelities. Design artefacts, as we make them available with INSPECTOR, can extend the designers' ability to collaborate more effectively and to develop a joint vision of a software product. Being logically linked, transitions from abstract to detailed artefacts increase the transparency of design decisions and enhance the traceability of dependencies on an electronic whiteboard. This helps to reveal errors, to dissolve contradictions, and improves communication, consistency, and lastly, the necessary understanding of the overall problem space that has to be made accessible through an innovative UI. Based on a ZUI approach, INSPECTOR integrates and innovatively interconnects the required artefacts in an interactive UI specification that provides good support for roundtrip engineering at any design stage. As thoughtfully selected artefacts in combination are more powerful [Brown et al., 2008], enhancing INSPECTOR in terms of collaboration (multi-modality), creativity (modelling and prototyping) and simulation ('living' specification) will make it an innovative and fully capable alternative to the tool landscape found in current industrial practice.

References

[Ambler, 2002] Ambler, Scott W.: Agile Modeling (John Wiley & Sons, NY, 2002)

[Ambler, 2004] Ambler, Scott W.: The Object Primer - Agile Model-Driven Development with UML 2 (Cambridge University Press, 2004)

[Barbosa et al., 2003] Barbosa S.D.J., Paula, M.G. (2003): Interaction Modelling as a Binding Thread in the Software Development Process, In Proc. of the workshop on bridging the gaps between software engineering and human-computer interaction, Oregon, USA

[Beck, 1999] Beck, K., Extreme Programming Explained (Addison-Wesley, 1999)

[Beyer et al., 1998] Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems (Morgan Kaufmann, 1998)

[Blomkvist, 2005] Blomkvist, S. (2005): Towards a model for bridging agile development and user-centered design. In: Human-centered software engineering – integrating usability in the development process, Springer, 219-244

[Brown et al., 2008] Brown, J.; Lindgaard, G.; Biddle, R. (2008): Stories, Sketches, and Lists: Developers and Interaction Designers Interacting Through Artefacts. In Proc. of the Agile Conference 2008, Toronto, Canada, 39-50

[Calvary et al., 2003] Calvary G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. (2003): A Unifying Reference Framework for Multi-Target User Interfaces. In Interacting with Computer 15(3): 289–308

[Campos et al., 2004] Campos, P., Nunes, N. (2004): Canonsketch: a User-Centered Tool for Canonical Abstract Prototyping. In Proc. of 11th International Workshop on Design, Specification and Verification of Interactive Systems, Springer: 146-163

[Campos et al., 2006] Campos, P., Nunes, N. (2006): Principles and Practice of Work Style Modeling: Sketching Design Tools. In Proc. of Human-Work Interaction Design. Springer

[Constantine et al., 1999] Constantine, L.L., Lockwood, L.A.D.: Software for Use: A Practical Guide to Models and Methods of Usage-Centered Design (Addison-Wesley, 1999)

[Holt, 2005] Holt, J.: A Pragmatic Guide To Business Process Modelling (British Computer Society, UK, 2005)

[Lepreux et al., 2006] Lepreux, S., Vanderdonckt, J., Michotte, B.. (2006): Visual Design of User Interfaces by (De)composition, In. Proc. of DSV-IS'2006, Vol. 4323, Springer, Berlin, 157-170.

[Lin et al., 2002] Lin, J., Landay, James A. (2002): Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. In Proc. of the 8th International Conference on Distributed Multimedia Systems, San Francisco: 573-580

[Memmel et al., 2007a] Memmel, T., Bock, C., Reiterer, H.. (2007): Model-driven prototyping for corporate software specification. In: Gulliksen, Jan (Eds.) Engineering Interactive Systems 2007 (EHCI-HCSE-DSVIS 2007), Salamanca, Spain

[Memmel et al., 2007b] Memmel, T., Gundelsweiler, F., Reiterer, H. (2007): Agile Human-Centered Software Engineering, In Proc. of the 21st BCS-HCI, Lancaster, UK, 167-175

[Memmel et al., 2007c] Memmel, T., Reiterer, H., Ziegler, H., Oed, R. (2007): Visual Specification As Enhancement Of Client Authority In Designing Interactive Systems. In: Kerstin Roese, Henning Brau (Eds.): Usability Professionals 2007, Frauenhofer IRB Verlag, Stuttgart, 99-104

[Memmel et al., 2008] Memmel, T., Reiterer, H. (2008): Inspector – Interactive UI Specification Tool. In Proc. of the 7th International Conference On Computer Aided Design of User Interfaces (CADUI) 2008, Springer, Albacete, Spain, 161-174

[Metzker et al., 2002] Metzker, E., Reiterer, H. (2002): Evidence-Based Usability Engineering. In C. Kolski, J. Vanderdonckt (Eds.), Computer-Aided Design of User Interfaces III, Kluwer Academic Publishers, Dordrecht, 323-336

[Newman et al., 2003] Newman, M. W., Jason, J. L., Hong, I., Landay, J. A. (2003): DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. HCI, 18(3): 259-324

[Nunes et al., 2004] Nunes, N. J., Campos, P. (2004): Towards Usable Analysis, Design and Modeling Tools. In Proc. of MBUI’2004, CEUR Workshop Proceedings, Vol. 103

[Rosson et al., 2002] Rosson, M. B., Carroll, J. M.: Usability Engineering: scenario-based development of human computer interaction (Morgan Kaufmann, San Francisco, 2002)

[Schrage, 1999] Schrage, M.:. Serious Play: How the World's Best Companies Simulate to Innovate (Harvard Business School Press, 1999)

[Shneiderman et al., 2006] Shneiderman B., Plaisant C. (2006): Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In Proc. of the 2006 AVI workshop on Beyond time and errors: novel evaluation methods for information visualization, Venice, Italy, 1-7

[Sousa et al., 2008] Sousa, K. S., Mendonca, H. et al. (2008): User Interface Development Lifecycle for Business-Driven Enterprise Applications. In Proc. of the 7th International Conference on Computer-Aided Design of User Interfaces CADUI’08

[Sutcliffe, 2005] Sutcliffe, G. (2005): Convergence or competition between software engineering and human computer interaction, In: Human-centered software engineering – integrating usability in the development process, Springer: 71-84

[Ware, 2004] Ware, C.: Information Visualization: Perception for Design (Morgan Kaufmann, 2004)

[Zave et al., 1997] Zave, P., Jackson, M. (1997): Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology 6, 1: 1-30