First publ. in: Proceedings of the Third IASTED International Conference on Human-Computer Interaction, March 17 - 19, 2008, Innsbruck, Austria / ed.: D. Cunliffe. Anaheim : Acta Press, 2008, pp. 170-179

INSPECTOR: METHOD AND TOOL FOR VISUAL UI SPECIFICATION

Thomas Memmel and Harald Reiterer
Human-Computer Interaction, University of Konstanz
Universitaetsstrasse 10, Box D 73, D-78457 Konstanz, Germany
{memmel; reiterer}@inf.uni-konstanz.de

ABSTRACT
When the user interface (UI) has to be specified, a picture is worth a thousand words and the worst thing a human-computer interaction (HCI) expert can do is attempt to write a natural language specification for it. Nevertheless, this practice is still common and it is therefore a difficult task to move from text-based requirements and problem-space concepts to a final UI design, and then back again. Especially for the specification of interactive UIs, however, HCI experts must frequently switch between high-level descriptions and detailed screens. In our research we found that advanced UI specifications therefore have to be made up of interconnected artefacts that have distinct levels of abstraction. With regards to the transparency and traceability of the rationale of the UI specification, transitions and dependencies must be visual and traversable. We introduce a UI specification method that interactively integrates interdisciplinary and informal modelling languages with different fidelities of UI prototyping. With an innovative experimental tool, we finally assemble models and design to a visual UI specification that will quickly take the place of text-based artefacts.

KEY WORDS
Usability Engineering, User Interface Development

1. Introduction

At the IASTED-HCI conference in 2007, we demonstrated how prototypes of different fidelity can contribute to UI design practice in agile usability-engineering processes [1]. We concluded that UI prototypes should be extended by interrelated, informal modelling languages. Consequently, we announced our idea of visual UI specifications that are intended to provide interactive access to both modelled artefacts and designs. Following up our previous research, we continued to put our approach into practice, and in this paper we present a method plus experimental tool-support for generating visual UI specifications. In Section 2 we provide an overview of the shortcomings of the common UI specification practice of many HCI experts. We then deduce in Section 3 the need for an advanced specification method and explain in detail how we identified adequate means for requirements modelling and UI design. Subsequently, we illustrate how we incorporated our findings in our visual UI specification tool, known as Inspector (Interdisciplinary UI Specification Tool). In Section 4 we exemplify our approach by going through a UI specification case study. In Section 5 we compare the contribution of our approach to related work, and we provide an outlook on our further research in Section 6. The paper closes with a summary and conclusion in Section 7.

2. Common UI specification methods

While approaches such as the usability maturity model (UMM) [2] or the DATech-Model [3] provide means to assess an organization's capability to perform HCI processes, they lack guidance aids on how to actually implement process improvement in HCI [4]. Consequently, many organizations lack modern UI specification and development processes that efficiently and effectively integrate HCI knowledge.

2.1 The dilemma of text-based UI specifications

In a survey of developers and business experts at Dr. Ing. h.c. F. Porsche AG, we - together with [5] - found that most HCI-related actors use text-based documents created in PowerPoint, Word or Excel for UI specifications (see Figure 1). The process is therefore visually awkward, which is very disadvantageous when designing UIs [6].

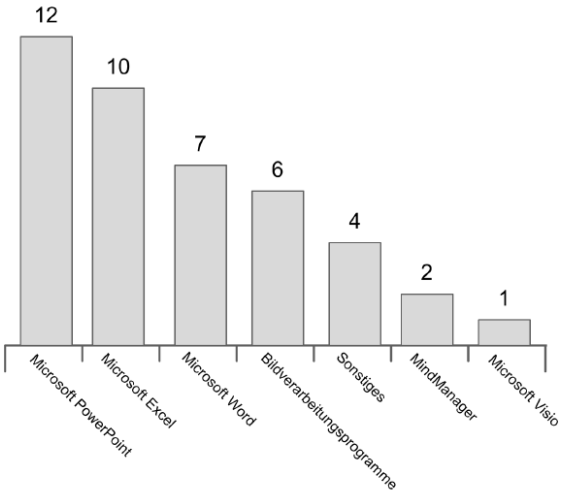

Figure 1: Office applications used for UI specification

There are several possibilities for error that can prevent text-based UI specification from being successful. First, different tools induce a mixture of formats and media disruptions. Second, programmers (i.e. the supplier) subsequently use completely different models and tools. They need to translate Office-like documents into more expressive models and even into code. Third, the black-box process of translating textual specifications into a running system makes it more difficult to cross-check the modelled UI against requirements. Fourth, text is

ambiguous and can easily become inconsistent with drawings that are maintained separately. There is a real danger of engineering failure, e.g. through misinterpretation or loss of precision. Fifth, text or loose images express the look, but not the feel, of the UI.

2.2 The problems with UML for UI specification

With modelling languages such as UML [7], software engineering (SE) tends to be more formal and model-based. For programmers, its diagrams are usually very helpful in producing a shared understanding and specifying backend system capabilities. The modelling language is clearly a step in the right direction, if only for the sole reason that software developers are no longer fighting the "notations wars" of the mid-1990s. HCI researchers with advanced SE expertise thus envision UML as being the appropriate vehicle for HCI experts as well. Various UML-based languages [8,9] have therefore been developed, but the extensions also have the disadvantage of leading to more formalism and decreased understanding for business people. In addition, developers probably need only 20 percent of the models to do 80 percent of the modelling work [10]. UML undoubtedly "is very strong at specifying the structure and the functionality of the application, (but) it is seldom used to (…) specify usability-related information" [11]. For UID issues, UML can only describe interactions formally and is unable to visualize real UI behaviour. In UML, use-cases do indeed concern users, but only in a fragmented way, in which each user goal is dealt with in isolation. As a result, HCI experts lose the global view of the application as it will be perceived by the users. This makes it difficult to envisage how the interaction will take place, which is essential in order to plan for, and evaluate, the quality of use of the final product. Consequently, and like text-based artefacts, UML is not adequate for specifying the look *and* feel of interactive UIs.

2.3 The missing intermediate solution

Previous research [12] reports that one year after introduction, 70% of CASE tools are never used, 25% are used only by a limited number of people in the organization, and 5% are widely used but not to capacity. Taking into account on the one hand the very informal UI specification practice we observed, and the inadequate formality and terminology of UML for UI issues on the other, we conclude that the most important reasons for avoiding professional UI tools are their poor usability [13], the great effort needed in learning to use them, and the difficulty of understanding the related terminologies. In a software market research we tested this hypothesis and analyzed 148 software tools for their practicability for UI design and specification. We were able to categorize the software applications into tools for requirements engineering (e.g. Axure Pro, Borland Caliber, IBM Rational, iRise Studio, Serena Composer, Telelogic Doors), for (formal) modelling (e.g. EclipseUML, iUML, MetaEdit+), for UI construction (e.g. Microsoft Expression Blend, Adobe Designer), for prototyping (e.g. Vision, SketchIt, FreeForm), for design (e.g. Adobe Photoshop, Corel Draw), and for animation (e.g. Adobe Flash, Adobe Flex). Afterwards, we mapped the software tools to important criteria for HCI experts (see Table 1). To accord with the capabilities of most HCI experts and actors in specification processes as described, an appropriate tool must avoid the need for coding in order to become popular and widely accepted, and to be used. Besides abstract prototyping, detailed simulations must be possible to evaluate UI behaviour. Modelling should provide room for informality and must be decoupled from UML to be applicable by HCI experts. Moreover, the deduction of design solutions from abstract models must be traceable and transparent [1]. From our analysis we concluded that only a very few UI tools currently support the HCI expert during UI specification tasks. Among them are, for example, Axure Pro, iRise Studio, Serena Composer and Sofeainc Profesy. These UI tools allow high-fidelity UI prototyping and facilitate simulations of the later system [1]. But whereas tools such as IBM Rational are bulked out with features far beyond the needs and capabilities of most HCI experts, in these otherwise promising UI tools the informal artefacts of early-stage HCI models, such as personas, essential use-cases, or task maps are entirely absent. Moreover, many existing UI tools are focused on the formalisms required to automatically generate the UI. This prevents current tools from adequately supporting the thought and design tasks that HCI experts have to accomplish in order to create usable, effective, and innovative UIs. All in all, we see no sign of any compromise between overwhelming HCI experts with features and formality on the one hand, and restricting their handicraft to UI prototyping on the other.

3. Visual UI specification with Inspector

Table 1: Main requirements for Inspector

Requirement	Effect
Moving from high-level descriptions of the UI to detailed screens	Easier transition from low-fidelity to high-fidelity UI designs
Designing different versions of, and changes to, a UI is easy and quick	HCI experts can build abstract and detailed prototypes rapidly
Allow an up-front usability evaluation of look and feel	The early detection of usability issues prevents costly late-cycle changes
Moving from business rules, use-cases and problem-space concepts into final solution design, and back again	Easier transition from problem space concepts to solution space; UI design can be back-traced with underlying models
Actors can build interoperable and simple models visually	All kinds of actors can pro-actively take part in the UI specification

A UI tool must not suffer from a "solve the whole problem" and "high threshold / low ceiling" illness. The

threshold is related to the difficulty of learning a new system, and the ceiling is related to how much can be done using the system [14]. The most important requirements for UI design and specification tools are traceability (switching back and forth between models) and smooth progression from abstract to concrete models [15]. This is backed up by the *Cameleon* reference framework [16], which structures the UI development into four levels of abstraction: task and concepts, abstract UI, concrete UI and final UI. These levels are structured with a degree of abstraction going from an abstract level to a concrete one and vice versa. In a work-style study, Campos and Nunes emphasised that the transition "moving from high-level descriptions of the UI (sitemaps, navigation maps, etc.) to detailed screens (with concrete widgets, buttons, etc.)" was the expert's most frequently executed task. In addition, the assignment to "move from business rules, use-cases and problem-space concepts into final solution (space) design, and back" was rated as the most difficult, and the second-most frequent, one [17]. As we will show in this paper, maintaining a connection between these levels of abstraction and changing perspectives from problem to solution space is well supported by our specification method and our corresponding experimental tool, known as Inspector. On the whole, we provide an approach that (1) covers all levels of abstraction from requirements gathering to UI specification, (2) gives the HCI expert more expressiveness throughout the process, (3) provides the chance to easily model UI-related artefacts, (4) supports UI design from sketching to exact specification, (5) is independent of specific technology, and (6) enables traceability and transparency across all integrated information. Finally, the outcome is a visual UI specification that defines the final system both in the abstract and in the detail.

3.1 Method: towards interdisciplinary UI modelling

HCI and SE are recognized as professions made up of very distinct populations. Each skill set is essential for the production of quality products, but no one set is sufficient on its own [18]. Modelling the UI in an industrial context also requires the integration of the discipline of business engineering (BE), which we define as the analysis, conception, and development of information systems as well as the relevant business-process re-engineering [19]. Typically, BE activities are driven by managers who mostly have a background in economics and marketing.

The interaction layer - as interface between system and user - is the area where HCI, SE and BE are required to collaborate. The HCI expert will be the one in charge of bringing it all together and ultimately he has to specify the UI. Hence, we bridge the disciplines with his point of view in mind. We extend his means of modelling and widen his perspective in terms of interoperable or supplementary artefacts of SE and BE.

As we found in our previous research, agile methods are close to HCI practice [1,20] and therefore the pathfinder for a course of action common to all 3 disciplines. Agile approaches already exist in SE [21] and HCI [20], and - with an agile version of the Rational Unified Process (RUP) - in BE [22] as well. RUP already includes some non-UML artefacts such as business rules or UI flow diagrams, whereas the other models can be simplified. Consequently, we can define a common denominator for all three disciplines (see Figure 2). Our goal is to keep this denominator as small as possible. We filter out models that are too difficult to be understood by every actor. Furthermore, we do not consider models that are more commonly used to support actual implementation (e.g. UML class diagrams). We integrate different levels of modelling abstraction to visualize the flow from initial abstract artefacts to detailed prototypes of the interaction layer (see Table 1). Figure 2 shows the resulting matrix of different models by discipline. On the vertical axis 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 initially 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. Consequently, we identify interoperable models at each stage as a common denominator for all three disciplines.

Text-based notations: personas and scenarios

For describing users and their needs, HCI recognizes user profiles, scenarios [23], role models [24], and personas [25]. Roles and personas are also known in SE and BE and are therefore appropriate for initial user modelling.

As an interdisciplinary modelling language, research suggests scenarios [26,27] - known as user stories (light-weight scenarios) in agile development [21]. 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. As interaction scenarios [23] are typically the most substantial, they can include all the described information and serve as the common denominator.

Graphical notations: usage and process modelling

Regarding graphical notations, we discovered similar modelling techniques across the disciplines. Although different in name, task cases (HCI), essential-use cases (SE), and business-use cases (BE) can all be expressed in a classical use-case notation. Moreover, use-case diagrams (SE, BE) overlap with use-case and task maps

(HCI) [24]. The latter also help to separate more general cases from more specialized (essential) sub-cases. We therefore offer both notations to the HCI expert.

We considered different models for task and process modelling. Following Ambler [10], we again selected related modelling languages. Activity diagrams 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, although they are more formal than the models HCI experts are familiar with. Sequence diagrams model the flow of logic within the system in a visual manner, enabling us to both document and validate the logic, and are commonly used for both analysis and design purposes. They focus on identifying the behaviour within the system. Sequence diagrams also model the logic of a usage scenario. As part of the business object model, they are one of most important design-level models for modern business application development [10]. Ultimately, we decided to maintain the expressivity of the original diagrams for the UI specification. Consequently, we identified three important (agile) models for this modelling level (see Figure 2).

For the storyboard layer we decided to keep the typical UI storyboards we know from HCI [25]. They are similar to UI flow diagrams (BE) [10] and can easily be extended to data flow diagrams. This means, for example, that we add different types of transitions and we indicate system back-end dependencies that are important for the UI.

All in all, the HCI expert has the opportunity to deal with some models he is normally unfamiliar with. But their application is made much easier if the agile versions of the models are used. With our modelling method, we give the HCI expert significant guidance in applying the right artefacts.

UI design with prototypes of different fidelity

Prototypes are already established as a bridging technique for HCI and SE [11]. Their role in SE was less important until agile approaches refocused attention on them as vehicles for inspections and testing, as well as a type of small release, which can be continuously changed and fine-tuned [28]. 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. They will help to design and evaluate the UI at early stages and they support traceability from models to design [1]. The visually most expressive level concerning the UI design is the high-fidelity prototyping layer. It provides a detailed model of the UI. It serves as the executable, interactive part of UI specification and makes the package complete. From here on, the actor can later 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.

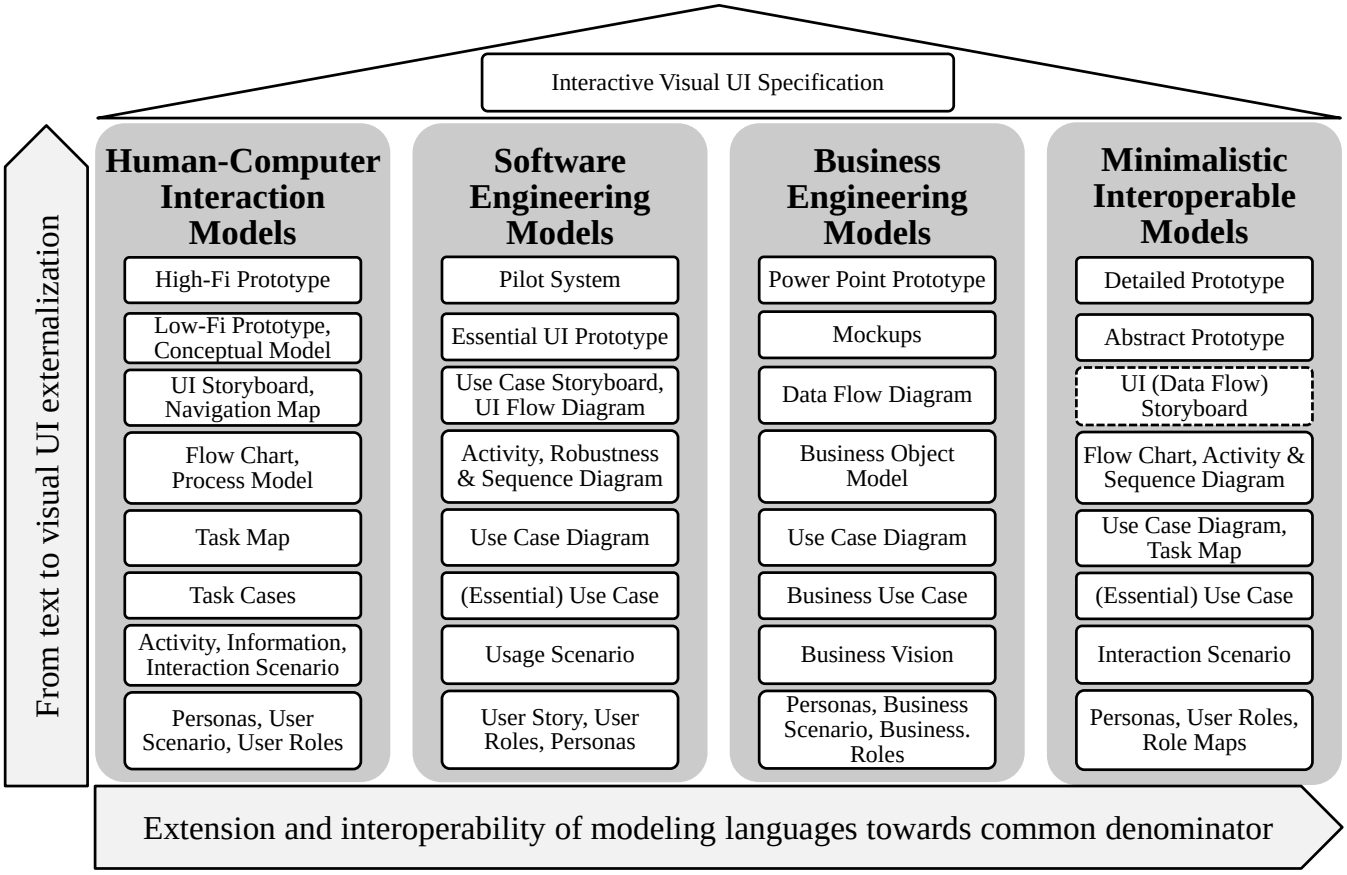

Figure 2: Interdisciplinary modelling languages for visual UI specification; storyboard as mediator of models and design

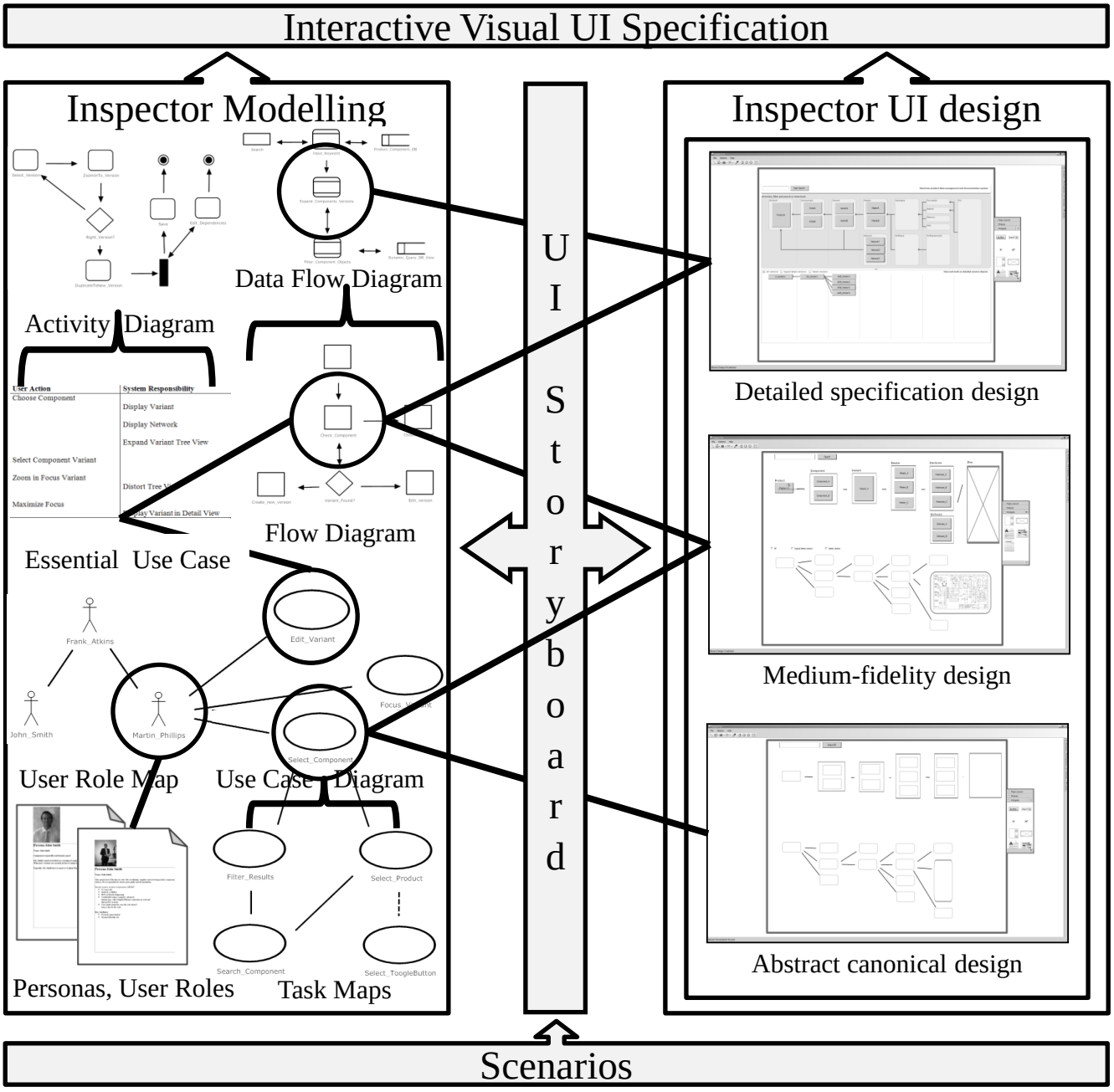

Figure 3: Correlation of models and UI designs as implemented with Inspector; exemplified modelling and design throughput

3.2 Tool: ZUI-based modelling and UI design

With regards to the hierarchy of interdisciplinary modelling languages (see Figure 2), we want to provide users with a feeling of *diving* into the information space of the UI specification. Because of our own experience and that of others in developing ZUIs [29,30,31], and due to related research on UI prototyping with ZUIs [32,33], Inspector offers panning and zooming as major interaction techniques [34]. 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). Automatic zooming automatically organizes selected objects on the UI. Animated zooming supports the HCI expert in exploring the topology of an information space and in understanding data relationships [35].

For switching between models and UI designs, the HCI expert can manually zoom in and out and additionally pan the canvas. Navigating between artefacts can be an extensive task, however, if the objects of interest are widespread in terms of being some distance along the three dimensions of the canvas (panning: x-axis, y-axis; zooming: z-axis). For a much faster change of focus, Inspector offers the possibility of creating links between models or elements of models. This is also the very important feature that supports traceability and transparency, as outlined in Section 3. Following such a link is executed as an animated jump zoom. Figure 3 shows how link operations (i.e. automated pan & zoom)

interconnect the models and designs created. As explained in Section 3.1, scenario descriptions are the initial model, whereas the UI storyboard functions as the mediator between modelling and design. At early stages, for example, a user shape can be linked to and be part of user roles, personas and use-case diagrams. 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. 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 to model the relationships of states, for example. The HCI expert can link every model to UI designs of different fidelity and vice versa. 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 [30] (see Figure 5). In addition, Inspector provides a tree-view explorer for switching between objects on the canvas. This navigation support allows a jump zoom into areas of the information space far removed from the current user focus.

4. Inspector: proof-of-concept study

By going through an outline of our UI specification efforts during an electronic product data management (EPDM) project, we want to further exemplify our approach. The general goals of EPDM are the reduction of documentation and change-management effort, the reduction of repair costs and effort and the reduction of risk with regards to international product liability. EPDM systems are applications that document and relate technical products and their subcomponents. A product typically consists of 1 to n components, and a component has 1 to n variants. The relationships between product-parts can be both logical and/or chronological and result in a corresponding complexity of versions. Chronological versioning of objects expands the data-space complexity by the dimension of time. The goal of the EPDM case study we describe in this paper is focused on improving usability for easier access to the information space. It is made with Inspector, which is based on Microsoft .NET 2.0 and currently consists of around 15,000 lines of C# code. Based on interviews with domain experts, we first described various scenarios with Inspector. The HCI expert can drag'n'drop the corresponding scenario shapes, represented as callouts, on the ZUI canvas (see Figure 4). By zooming into them, the callouts can be enriched by direct text input or the integration of external documents such as Word or PDF. Every callout can contain from one to many scenarios with different purposes (e.g. activity scenarios, information scenarios, interaction scenarios). The scenarios are used to identify threads of usage for the system to be constructed and to provide a description of how the system will be used [23]. We also used them to consider adequate means for information visualization, exploration, and search (see Table 2), and we incorporated suitable definitions and example images.

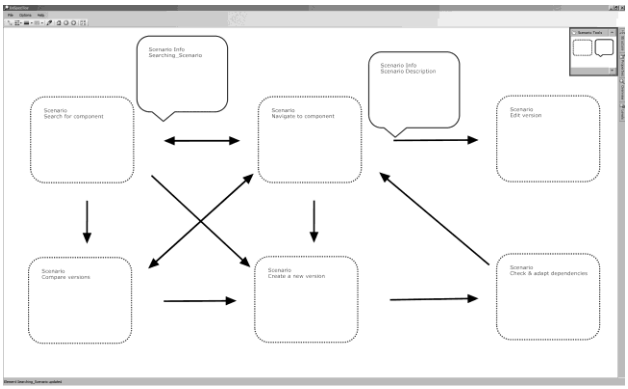

Figure 4: Scenario level with scenarios and aggregated tasks

With regard to information visualization (1), we identified a need to provide both overview + detail. The overview must provide the "big picture" of product-part relationships. The detail must visualize the interdependencies of versions. A drill-down into detail will reveal more of the required product data. At present, navigation (2) in EPDM systems is largely based on an expandable, sometimes redundant, hierarchical tree representation, although data is rather highly networked. Consequently, users lose their overview of product-part relationships. We need to give users a permanent clue as to where they are, where they can go, and where they have already been. With regard to search (3), instruments for narrowing the information space with filter options are necessary.

Table 2: General requirements for the UI of EPDM

#	Topic	Requirement for UI
1	Information Visualization	Visualize information space and relationships with overview-plus-detail techniques; let users define the degree of visual complexity
2	Navigation	Provide advanced navigation concepts; maintain context of use
3	Search	Provide search & filter techniques

Table 3: Typical use-cases relevant in EPDM

#	Task Description
1	Navigate to component and versions
2	Search for component and versions
3	Create a new component version
4	Check and adapt dependencies of versions
5	Compare different versions
6	Edit an existing version object

Consequently, we then associated the scenarios with different, abstract main tasks (see Table 3) that the system needs to support. They are represented by rounded rectangles on the Inspector canvas (see Figure 4). If necessary, the HCI expert can already highlight

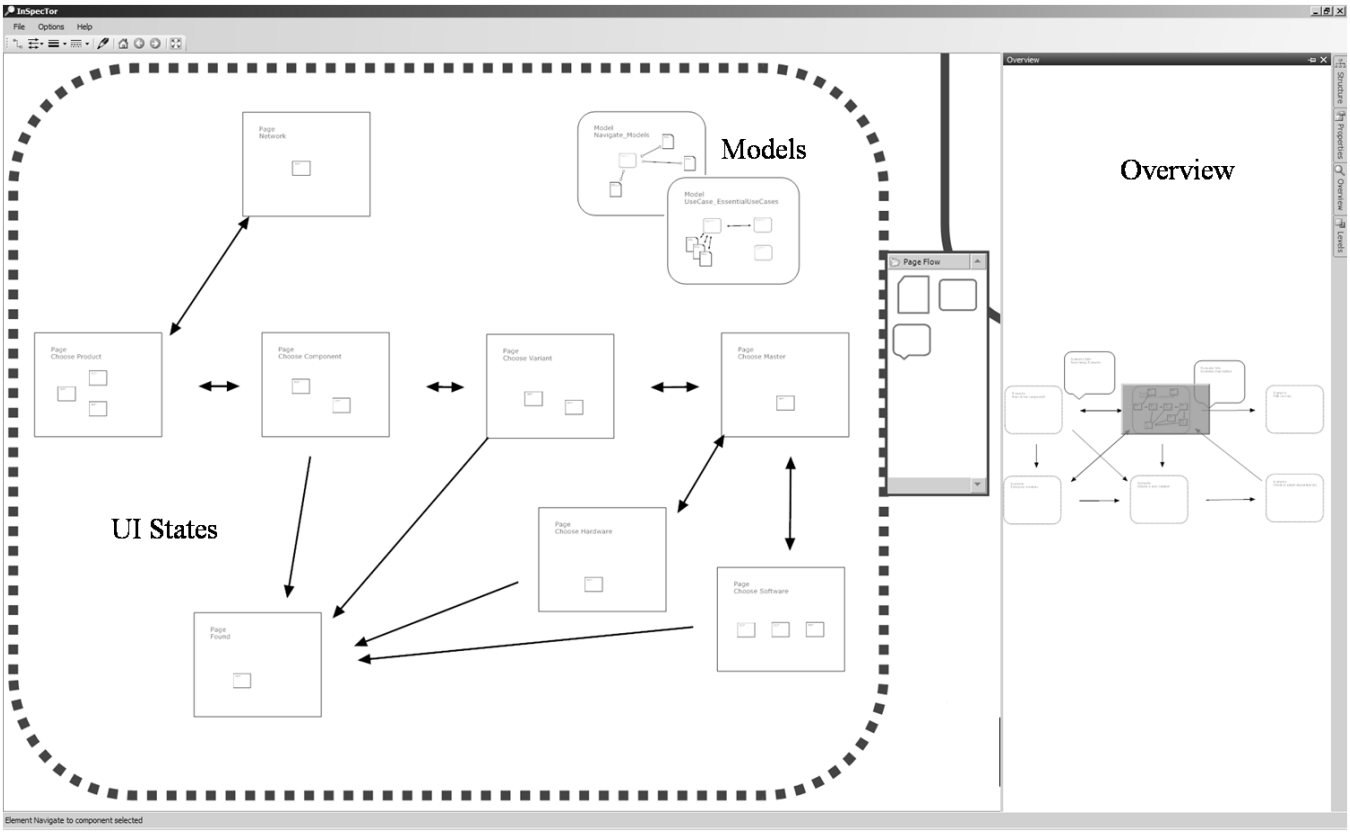

Figure 5: Storyboard for UI states (bottom left) and related models (centre top); overview (right) as navigation aid

interdependencies among abstract tasks with different kinds of arrows. Whereas the callouts contain the scenario descriptions, the abstract tasks function as grouping objects [24] for detailed modelling and UI design. On clicking them, the HCI expert triggers an automatic zoom that centres the maximized shape on the canvas, and the shape reveals its detailed content. Inspector then provides areas and tools adequate for the task of storyboarding (semantic zoom). On the one hand there are areas for abstract and detailed UI design, and on the other there are shapes for modelling the requirements (see Section 3). The storyboard view shows where the HCI expert has already modelled content. The UI design will later function as an interactive part of the specification that can, for example, be used to evaluate the interface. Transitions between UI representations will be shown on the storyboard automatically as soon as the HCI expert creates links that induce them. The models provide additional information that is only indirectly expressed at the UI layer, such as underlying use-cases or addressed personas (see Section 3.1). Our UI specification method typically starts by modelling users and their tasks (see Figure 3). Accordingly, the HCI expert uses a modelling shape to integrate the requirements (see Figure 5). As explained in Section 3.2., the HCI expert can connect different models to each other and connect models to UI designs. In Figure 6 we created a use-case diagram and linked the user shape to the corresponding personas model. The use-case shapes are connected to the relevant essential use-case descriptions. The overall use-case diagram is linked to a group of UI representations (or versions of them) on the storyboard layer (see Figure 5). As soon as basic requirements have been defined, the HCI expert can begin to externalize initial ideas about the UI. At the storyboard layer, the HCI expert can first create placeholders for UI states and then start to model them in more detail. Each UI state captures versions of its UI designs and also allows annotations in callouts for them. Thus, the rationale of different UI designs is stored. To edit a design, the expert can double-click (goal-directed, semantic zoom) or use the mouse-wheel to smoothly zoom in on the focused shape.

Inspector provides tools and widgets for low- to high-fidelity prototyping. Accordingly, the tool offers features for sketching or using customizable geometric shapes familiar to most actors from office applications. This activity is also related to canonical abstract prototyping [14], as the vague design of UI elements will later be enhanced by innovative interaction controls. The geometric shapes can be freely scaled and configured in their appearance (border colour, fill colour, etc.). Predefined template shapes can be dropped on the canvas and can be used as backup for the UI design. Naturally, low-fidelity prototyping can become medium-fidelity, as the expert has the freedom to add images or more concrete widgets if needed. For more sophisticated and detailed designs, Inspector – as a high-fidelity UI builder - provides common UI widgets and functions (see Figure 7). At any level of UI detail, external objects such as images or even ActiveX components can be embedded on

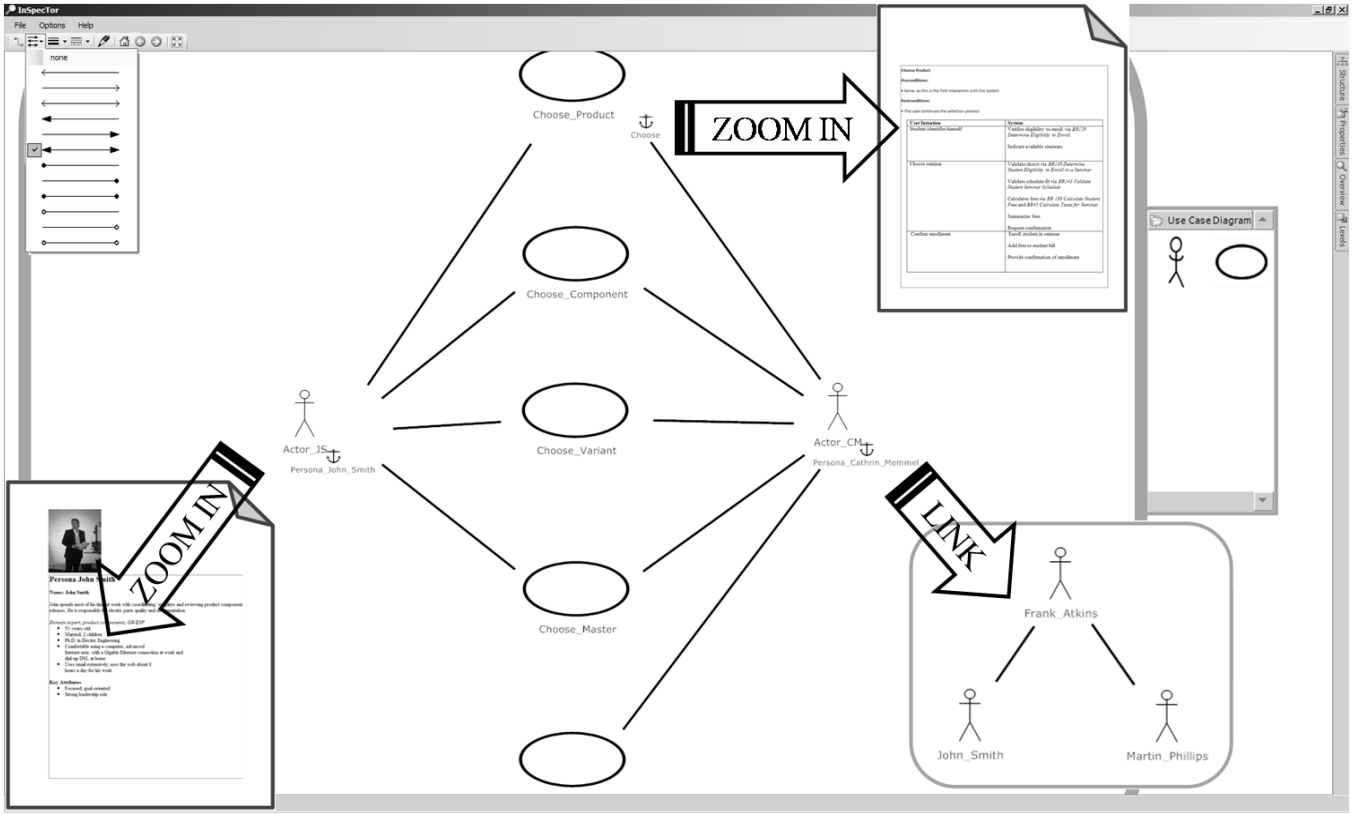

Figure 6: Use-case diagram with zooming from actor to personas (left), case to essential use-case (top), & link to role map

the canvas. Importing images is helpful in situations where existing resources, or parts of them, are reused. Especially during later stages of design, the HCI expert may want to integrate mockups of UI elements or whole screens. ActiveX is helpful when specific widgets have been prepared in advance; for example, by programmers. In general, later on this will provide the necessary guidance for the programmers who actually code the final UI. As part of an interactive UI specification, the UI designs being modelled must be connected to a flow of UI states. With Inspector, UI elements can be set up to point to other states. This is possible, for example, by drag'n'dropping another UI state from the supplementary object explorer onto the widget (see Figure 7). Inspector then creates a link between the states and the link is also visible on the storyboard (see Figure 5). Consequently, the HCI expert can evaluate an interactive simulation of the UI design. Moreover, and as explained in Section 3, UI states can also link to underlying models. In general, Inspector can thus be used as a browser for the UI specification. All artefacts that have been created can be saved to an XML file and exchanged with actors.

5. Related Work

Campos [14] presented the tools CanonSketch and TaskSketch. CanonSketch supports abstract UI prototyping and is based on the concept of canonical abstract prototyping. TaskSketch supports collaborative modelling, but focuses on essential use-cases only. Although both tools therefore share some basic concepts with our approach, when using Inspector, the HCI expert can choose from a wider range of means of expression, both on a more detailed level and on some levels that are more abstract than essential use-cases. Our approach is more comprehensive and flexible in terms of utilisable artefacts. Moreover, Inspector assembles models and designs into a tangible, visual UI specification that can be experienced. Instead of dealing with selected stages of UI design, we provide a sound process model.

6. Evaluation & Further Research

During the EPDM project, we were able to successfully introduce Inspector as a new means of requirements elicitation and UI specification. We will extend Inspector by activity modelling that allows superordinate activities to be related to tasks. Activity models will then replace the general task shapes we use on the scenario level (see Figure 4). They will also induce a temporal sequence of user activity. Moreover, we want to merge data-flow diagrams with the UI storyboard more systematically. This will further reduce the number of different models. We will then conduct a lab usability study to analyze Inspector, especially with regard to its interaction mechanisms. Moreover, we plan a review with the HCI experts we initially interviewed in order to evaluate the completeness and maturity of our specification method.

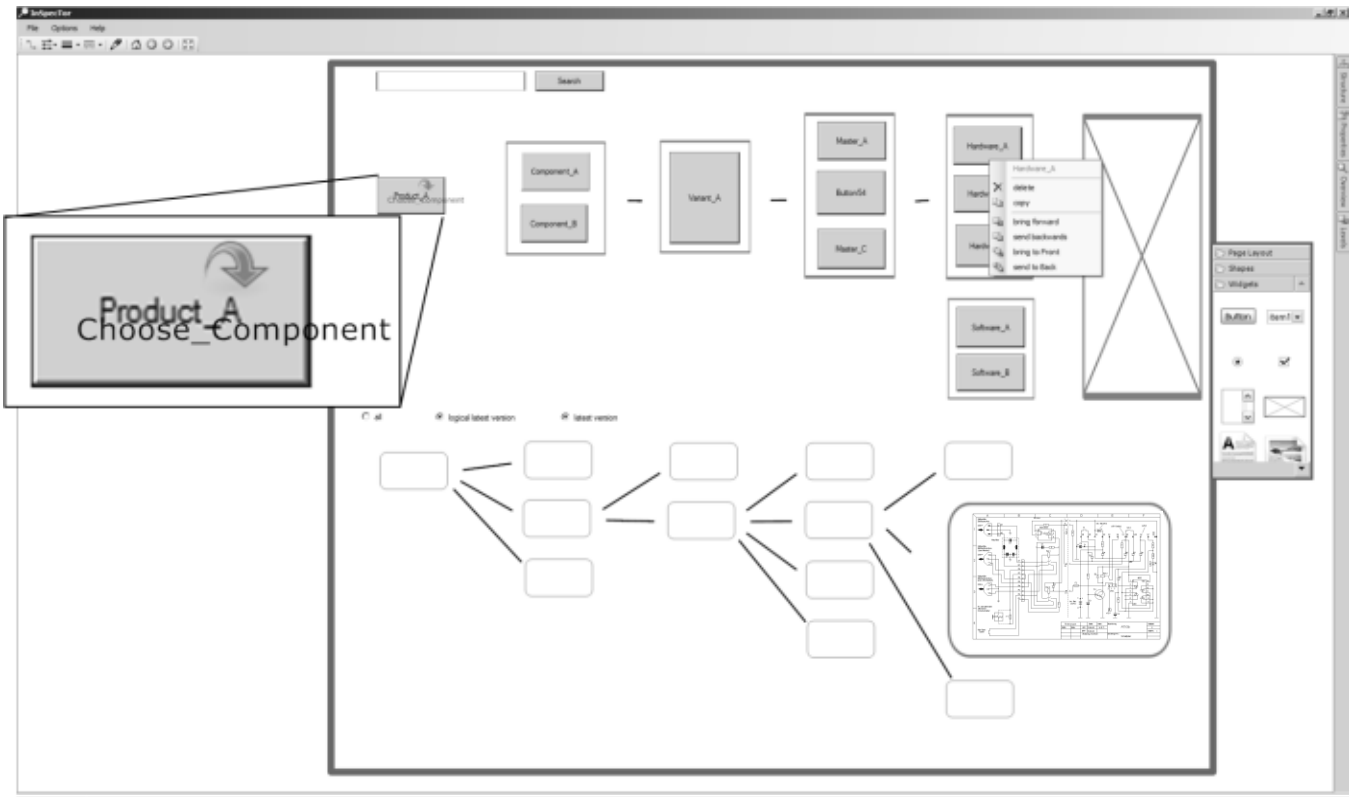

Figure 7: UI with mixed fidelity; linked button (top left) canonical elements, concrete widgets, pre-designed pictive elements

7. Summary & Conclusion

Based on our experience in UI specification and design, we have come to the conclusion that the typical methods and tools available to HCI experts are not adequate. UI tools must support not only the "hard" aspects, but also the "soft" aspects of UI development to support the delivery of usable systems in the future. These include support for creativity and improvisation. Text-based, abstract specification processes are a stumbling block for HCI experts. With our method and experimental tool-support, HCI experts should be well supported in applying informal models and prototyping with different fidelities. Being logically linked, transitions from abstract to detailed artefacts increase the transparency of design decisions and enhance the traceability of dependencies. This 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, our Inspector tool integrates and innovatively interconnects the required artefacts in a visual UI specification that can be interactively experienced. With our approach, we focus on HCI experts in charge of the conceptualization, and particularly the specification, of UIs. We therefore neither suggest a CASE-tool, nor do we envision the automatic generation of the final UI. We propose a method and a tool as connecting factors to encourage HCI experts to reconsider common UI specification practice.

References

[1] T. Memmel, F. Gundelsweiler, H. Reiterer, Prototyping Corporate User Inter-faces – Towards A Visual Specification Of Interactive Systems. *In Proc. of the IASTED-HCI 2007*, Acta Press, 2007, 177-182

[2] J. Earthy, Human Centred Processes, their Maturity and their Improvement. *In Proc. of IFIP TC.13 International Conference on Human-Computer Interaction*, 1999, 117-118

[3] DATech: Qualität des Usability Engineering Prozesses, Deutsche Akkreditierungsstelle Technik e.V., 2001

[4] E. Metzker, H., Reiterer, Evidence-Based Usability Engineering. *In Proc. of the Computer-Aided Design of User Interfaces conference*, 2002, 323-336

[5] C. Bock, D. Zühlke, Model-driven HMI development – Can Meta-CASE tools relieve the pain? *In Proc. of the First International Workshop on Metamodelling – Utilization in Software Engineering (MUSE)*, Portugal, 2006, 312 319

[6] I. Horrocks, *Constructing the user interface with statecharts* (Addison-Wesley, Harlow, 1999)

[7] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (Addison-Wesley, 2003)

[8] M. Abrams, C. Phanouriou, UIML: An XML Language for Building Device-Independent User Interfaces. *In Proc. of the XML '99*, Philadelphia, 1999

[9] P. P. da Silva, N. W. Paton, User Interface Modeling in UMLi. *In IEEE Software, 20*(4), 2003, 62-69

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

[11] A. G. Sutcliffe, Convergence or competition between software engineering and human computer interaction, In: Seffah, A., Gulliksen, J., Desmarais, M. C. (eds.), *Human-centered software engineering – integrating usability in the development process*, Springer, 2005, 71-84

[12] C. F. Kemerer, How the Learning Curve Affects CASE Tool Adoption. *In IEEE Software 9*, 1992, 23-28

[13] S. Jarzabek, R. Huang, The case for user-centered CASE tools, *In Communications 41*(8), ACM Press, 1998, 93-99

[14] P. Campos, N. Nunes, Canonsketch: a User-Centered Tool for Canonical Abstract Prototyping. *In Proc. of DSV-IS'2004, 11th International Workshop on Design, Specification and Verification of Interactive Systems*, Springer-Verlag, 2004, 146-163

[15] H. Trætteberg, P. J. Molina, N. J. Nunes, (eds.): *In Proc. of the IUI/CADUI'04Workshop on Making model-based user interface design practical: usable and openmethods and tools*, Portugal, 2004.

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

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

[18] B. Baxton, Performance by design: The role of design in software product development. *In Proc. of the Second International Conference on Usage-Centered Design*, Portsmouth, 2003, 26-29

[19] Y. Malhotra, Business Process Redesign: An Overview, *In IEEE Engineering Management Review, 26(3),* 1998

[20] T. Memmel, T., F. Gundelsweiler, H. Reiterer, Agile Human-Centered Software Engineering, *In Proc. of the 21st BCS-HCI*, 2007, 167-175

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

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

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

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

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

[26] S.D.J. Barbosa, M.G. Paula, 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. 2003

[27] M. Jarke, Scenarios for modeling, *Communications of the ACM, 42*(1), 1999, 47-48

[28] S. Blomkvist, S., Towards a model for bridging agile development and user-centered design. In: Seffah, A., Gulliksen, J., Desmarais, M. C. (eds.), *Human-centered software engineering – integrating usability in the development process*, Springer, 2005, 219-244

[29] J. Raskin, *The Humane Interface - New Directions for Designing Interactive Systems* (Addison-Wesley, 2000)

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

[31] H.-C. Jetter, J. Gerken, W. A. König, C. Grün, H. Reiterer, HyperGrid - Accessing Complex Information Spaces, *Proc. of the 19th British HCI Group Annual Conference*, Springer, 2005

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

[33] M. W. Newman, J. L. Jason, I. Hong, J. A. Landay, DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. *In Human-Computer Interaction*, *18*(3), 2003, 259-324

[34] K. Perlin, D. Fox, Pad: An alternative approach to the computer interface. *Proc. of the 20th Annual ACM Conference on Computer Graphics*, J. T. Kajiya, Ed. ACM Press, New York, N.Y., 1993, 57–64

[35] B. B. Bederson, A. Boltman, Does Animation Help Users Build Mental Maps of Spatial Information? Tech Report CS-TR-3964, Computer Science Department, University of Maryland, College Park, MD, 1998