In: Proceedings of *4th European Conference on Interactive Television* (G. Doukidis, K. Chorianopoulos and G. Lekakos, eds., 2006), ELTRUN-Athens University of Economics and Business, pp. 73-82.

Design Methods for interactive TV: two empirical studies

Michel Alders[1], Reinder Haakma[2], Matthias Rauterberg[3]
([1]) TomTom BV, ([2]) Philips Research Eindhoven, ([3]) Eindhoven University of Technology; The Netherlands
([1]) michel.alders@tomtom.com ([2]) reinder.haakma@philips.com ([3]) g.w.m.rauterberg@tue.nl

Abstract
The central question for this paper is how to improve the production process by closing the gap between industrial designers and software engineers of television(TV)-based User Interfaces (UI) in an industrial environment. Software engineers are highly interested whether one UI design can be converted into several fully functional UIs for TV products with different screen properties. The aim of the software engineers is to apply automatic layout and scaling in order to speed up and improve the production process. However, the question is whether a UI design lends itself for such automatic layout and scaling. This is investigated by analysing a prototype UI design done by industrial designers. In a first requirements study, industrial designers had created meta annotations on top of their UI design in order to disclose their design rationale for discussions with software engineers. In a second study, industrial designers assessed the potential of four different meta annotation approaches. The question was which annotation method industrial designers would prefer and whether it could satisfy the technical requirements of the software engineering process. One main result is that the industrial designers preferred the method they were already familiar with, which therefore seems to be the most effective one although the main objective of automatic layout and scaling could still not be achieved.

1. Introduction

This paper is about the investigation of a part of the product creation process of television (TV)-based user interfaces (UIs) in a multinational of consumer electronics. Two parties are involved in this process: (1) industrial designers with highly professional design skills, and (2) software engineers relying on formal specification methods. Designers create the UI concept, which they make concrete by delivering a partial prototype: a UI design. Software engineers have to convert the UI design into a fully functional software product running on diverse hardware platforms.
Several aspects are at stake in the product creation process (Rauterberg et al., 1995) (Rauterberg, 1996), and one of them is an effective graphical layout (the look) of the interaction structure. A lot of work has already been published on the graphical layout of information objects (Mackinlay, 1986) (Mackinlay, 1988) (Mackinlay, 1991) (Kamps et al., 1994), unfortunately not for interaction structures. In order to improve the implementation of the graphical layout of the interaction structure, the software engineers at the research centre of a multinational company explored whether one UI design can be converted into several functional UIs for hardware platforms that have different screen properties. Their aim is to apply automatic layout and scaling of the UI in order to improve the product creation process (Vanderdonckt, 1994). However, the question is whether a UI design lends itself for such automatic layout and scaling. This is investigated by analysing a prototype UI design with the help of professional industrial designers who created it (Rauterberg et al., 1995) (Rauterberg, 1996) (Bødker et al., 2000).
In our first study, the industrial designers clarified the UI design by creating annotations on top of it. Since annotations can also improve the product creation process, a second study was carried out in order to investigate the potential of four different annotation methods. This formative approach (Vincente, 2000) (Atwood et al., 2002) is taken in order to assess how the most import meta information can be transferred from industrial design to software engineering easily and reliably (von Knethen et al., 1998).
Annotations can reveal the rationale behind the UI design, and the explicit design rationale can support software engineers to go beyond merely using the UI design provided via a prototype in order to formulate the underlying layout algorithms. Annotations also have the potential to limit the occurrence of inconsistencies that a UI design and its later implementation can contain (Kamsties et al., 2001).
The main question is how industrial designers feel about using a semi-formal annotation method and which type of annotation method they would prefer as they will be required to use them in their UI design process (Atwood et al., 2002). This is an important aspect because creative industrial designers almost feel threatened by formal specification methods. Any [semi-]formal annotation method is perceived as constraining the expressive power of the design space. This paper will describe two empirical studies and discuss them in the context of the product design process for interactive TV. A general discussion and the conclusions will finalise this paper.

Figure 1: Screenshots of the UI design used in this study, showing the main menu (left) and an example of a list widget (right).

2. Study 1: Analysis of a UI Design

In our first study the layout of a prototype UI design is analysed in order to check whether software engineers can convert it into several fully functional UIs for products with different screen properties.
The software engineers aim for automatic layout and scaling in order to improve the product creation process and bridge the gap between software and design. Ideally, software engineers use one UI design and convert that into one fully functional UI that can run on different products with different screen properties. Such UIs should meet different requirements resulting from diversity issues. There is diversity between products as a result of different screen properties, for example, PAL versus NTSC, 4:3 versus 16:9, dual screen and iPronto. Also within the layout of a single UI diversity can occur. Menu labels for instances, consist of a label background and a text string that is positioned on top of it. The text string should never exceed the length of its background, but can itself become longer due to a translation or because a file name is exceptionally long (e.g. DVD file names). Automatically scaling the label background and adjusting the layout accordingly could deal with such diversity.
The question is to what extent a current UI design is able to take diversity issues into account in order to support automatic layout and scaling. Furthermore, the aim of this study is to reveal the implicit guiding principles designers use in their work (Kurosu et al., 1995) (Tractinsky, 1997) (Karvonen, 2000).

2.1 Approach

Starting point of this study was a particular UI design (given as a partial prototype) for interactive TV that has been developed with Macromedia Director. This UI design is a typical TV-UI because similar UIs can be found in many current TV products (see Fig. 1). The UI design shows in full detail only a selection of all possible widgets, i.e. the menu (see Fig. 1; left) and a list widget (see Fig. 1; right). The functions have fictitious names (e.g. Item 1 stands for 'Picture', Item 2 stands for 'Sound', etc). A series of interview sessions was conducted with two experienced industrial designers. During seven sessions of about four hours each, they were asked about the rationale behind position and size of the UI elements and about the implicit guidelines they use. After each session, the designers incorporated their clarifications in Director as additional information layers on top of the UI design. This annotation method is from now on called the Director method.

2.2 Results

The designers have told that all the graphical elements in the UI design are positioned and sized in absolute, and not in relative terms.
Regarding the length of text strings the designers pointed out that they would never allow the label background to adjust its size automatically to the size of the text string. If the length of the string is known at design time (e.g. menu labels), they rather adjust the size of the label background to the longest string and use that length of the label background throughout the UI design. Abbreviations can also be used to fit a (translated) string properly on a label background. Text strings that are not known at design time (e.g. DVD file names) do not occur in current TV UIs. Designers explained that if such strings would occur, they would make use of abbreviations or the next line (as in text documents).
The most important guidelines for designers stem from typographical rules such as readability. Another important guide for their work is a company internal UI standard. However, this standard focuses on the user interaction. Little guidelines are provided about the look or layout.
The end result of the seven meetings with the designers was an annotated UI design. The designers

created six additional information layers on top of their original UI design. The six annotation layers, partly shown in Fig. 2, Fig. 3 and Fig. 4, consist of:

- *Elements*: name tags for the UI elements.
- *Safe area*: indication of an area inside the screen borders of 50 pixels, where important UI information can be placed in order to be properly perceptible.
- *Menu layout grid*: guides for aligning (animating) UI elements (see Fig. 2, left).
- *Widget layout grid*: placement of the elements a widget consists of (see Fig. 2; right).
- *Widget grid*: alignment within the widget, animation path included (see Fig. 3).
- *Dimensions*: detailed alignment of a UI element (e.g. a label), animation path included (see Fig. 4).

2.3 Discussion

In order to support automatic layout and scaling for different screen properties, the UI elements need to be positioned and sized in relative terms (Marriott et al., 2002). However, in current practise, designers position and size UI elements only in absolute terms. Label backgrounds are also not positioned and sized relatively. Therefore the resulting UI design cannot be converted into a reusable UI for different screen properties. In this study, the industrial designers used an annotation method (the above called Director method) in order to clarify the UI design and shed light on their design rationale. This Director method shows that the designers mainly use grids and guides in order to position UI elements on the screen. Grids and guides can act as a constraint. It enables software engineers to formulate layout algorithms that are consistent throughout the UI. For instance, the detailed internal grid (see Fig. 4) is used to position a text string on a label background consistently. As such, a group is created (a label). The constraints of such a group ensure that it can easily be reused in several places within the UI and therefore supports consistency. Typically, inconsistencies arise due to several people working on the same project (Rauterberg et al., 1995) (Kamsties et al., 2001). The Director method does not take into account different screen properties. Therefore this method is incomplete in that it is not specified how the grids and guides should be repositioned when the screen properties change. In our second study the Director method will be compared with three other annotation methods.

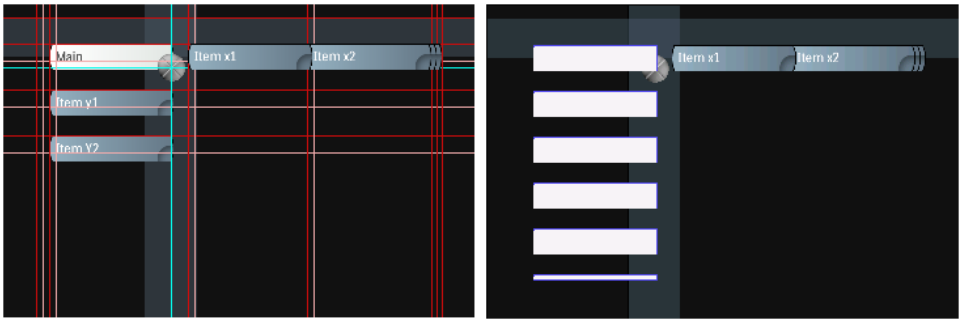

Figure 2: Screenshots of the menu layout grid (left) and the widget layout grid (right).

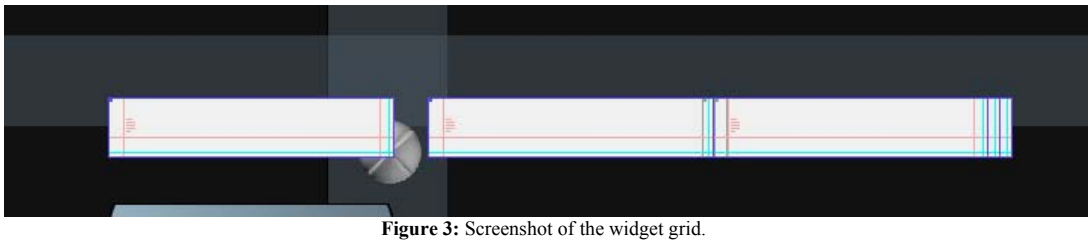

Figure 3: Screenshot of the widget grid.

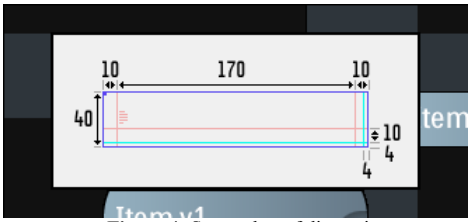

Figure 4: Screenshot of dimensions.

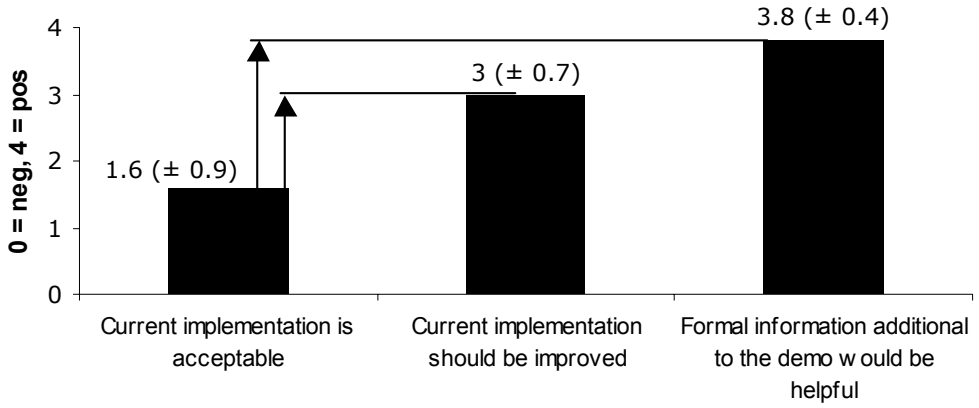

Figure 5: Evaluation of current implementation and the need for formal additional information (N=5). Arrows designate which answers differed significantly in a post-hoc comparison (Scheffé).

3. Study 2: Four Different Annotation Methods

At the outset of our second study into annotation methods for UI design, a short questionnaire amongst five industrial designers was performed (5 point rating scale: 0='negative' to 4='positive'). It revealed that using formal information additional to the interactive prototype (i.e. an annotation method for UI design) could be helpful (see Fig. 5). A MANOVA (repeated measure; SPSS version 11.0 for Windows) showed that the 3 questions differed significantly ($F=34.57$, $df=1$, $p=0.004$). Also, a post-hoc comparison was performed in order to determine which answers differed significantly (indicated by arrows, see Fig. 5). These promising results motivated us to investigate the possibility of introducing an annotation method in the UI production process further on.

In order to assess the advantages and disadvantages of the Director method described in our first study, it is compared with three other annotation methods. The aim of the assessment study is to identify how designers feel about producing the proposed annotation methods, because they should provide such information in the product creation process. Further, it should become clear which annotation method they would prefer.

Annotations shed light on the implicit design rationale and can therefore limit the occurrence of misinterpretations and possible inconsistencies. Inconsistencies can arise due to several people (designers as well as software engineers) working on the same project (Rauterberg et al., 1995). Ideally, a standard annotation method comes forward that points out rules and constraints about position and size of the UI elements in order to support software engineers to formulate the underlying layout algorithms, also with respect to different screen properties.

3.1 Approach

The four alternative annotation methods that are compared in this study are called the (1) 'Spreadsheet' method, (2) the 'Screenshot' method, (3) the 'Director' method and (4) the 'Construction Tool' method. Only the Director method (3) is produced by designers themselves. All the three other methods are introduced for the purpose of this second study. In general the four annotation methods are comparable in that designers should be able to use them with widely available high-level tools (e.g., Excel, Drawing tools, etc). All annotation methods aim to minimize the occurrence of inconsistencies.

The annotation methods differ with respect to the extent that the design rationale can be expressed in order to support software engineers. Also to what extent automatic layout and scaling is supported differentiates the four annotation methods. Below the four annotation methods are outlined.

The 'Spreadsheet' method: This form of annotation is created in Excel and consists of an enumeration of all the UI elements ('Object') that occur in the UI design and their width (pixels), height (pixels), colour and transparency. Table 1 shows a part of such an annotation table.

This Spreadsheet annotation solely provides information about the UI elements in absolute terms as they occur in the UI design. Therefore, the information is incomplete in that it is not specified what should happen with the size and position of the elements if the properties of the screen change. Furthermore, it does not provide rules or constraints such as the maximum length of a text string. Therefore, this method is less appropriate to express explicitly the design rationale than any other method.

Table 1: Example of the Spreadsheet method. Described is the make up of the background bars as they occur in the UI design.

Object	Width	Height	Colour	Transparency
Vertical menu background_1	1	480	Black	0%
Vertical menu background_2	79	480	Black	50%
Horizontal menu background	720	60	Black	50%

The 'Screenshot' method: The Screenshot method consists of a set of screenshots that are annotated by making notes on top of it (with a simple drawing tool) (Keränen et al., 2000). The position, size and transparencies of the UI elements are explained. In order to explain position and size of UI elements as many screenshots can be annotated as needed. Furthermore, UI elements are described apart from their context as is shown in Fig. 6.

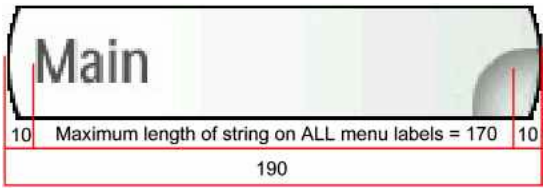

Figure 6: A part of the UI is highlighted in order to explain a rule: the maximum length of a text string and its label background.

Typically, an explanation consists of a written comment that refers to the UI elements in question by making use of arrows and the like. For example Fig. 7, regarding the selection of a horizontal menu item (here 'Item x1'), three rules are stated that apply to ALL horizontal menu items. This method also contained one example regarding different screen properties. It states what happens with horizontal menu items when the horizontal screen space is limited. This method is able to express the design rationale easily by specifying rules and constraints in textual form. It provides only limited information about what should happen when the properties of the screen change. In practise though, it depends on the user (the writer) of this method how much rules and constraints are going to be specified.

The 'Director' method: The Director method is described in detail in the results of our first study (see chapter 2.1 above). It specifies constraints in the form of grids and guides in order to express the design rationale. However, the information is incomplete in that it is not specified how the grids and guides should be repositioned when the screen properties change.

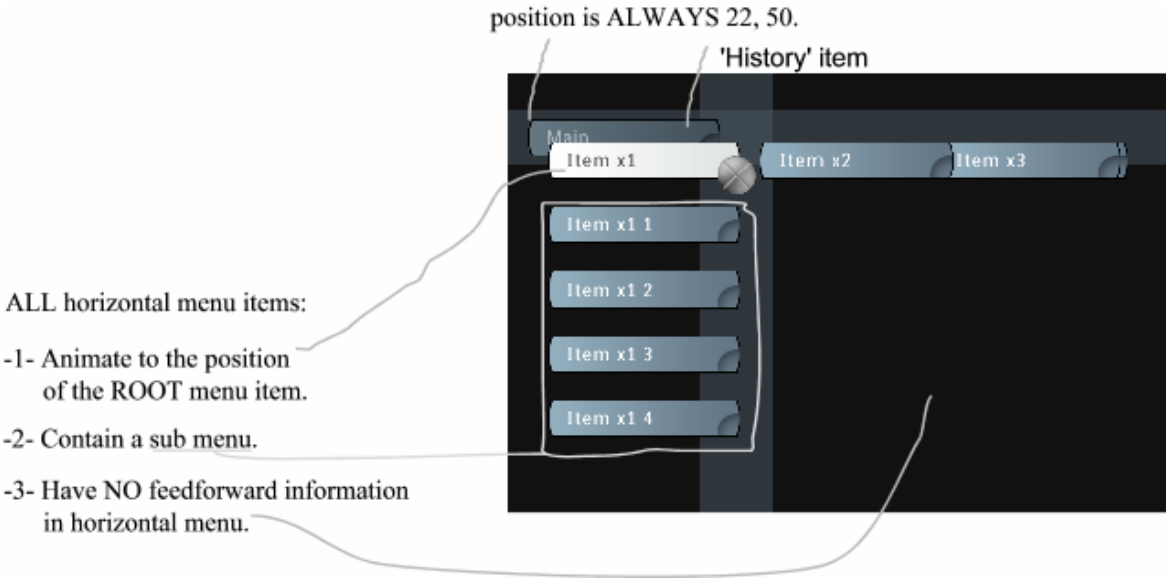

Figure 7: Example of the Screenshot method where the selection of a horizontal menu item is explained by annotations.

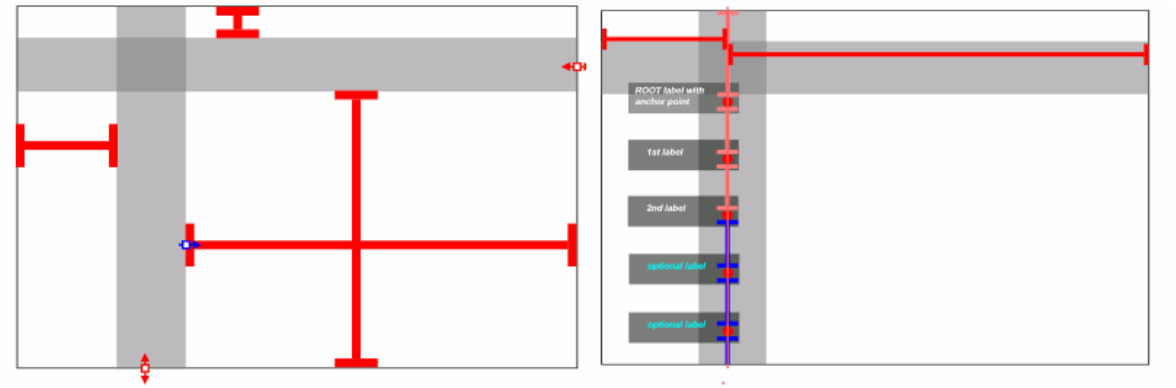

Figure 8: Two examples of the Construction Tool method (see text above).

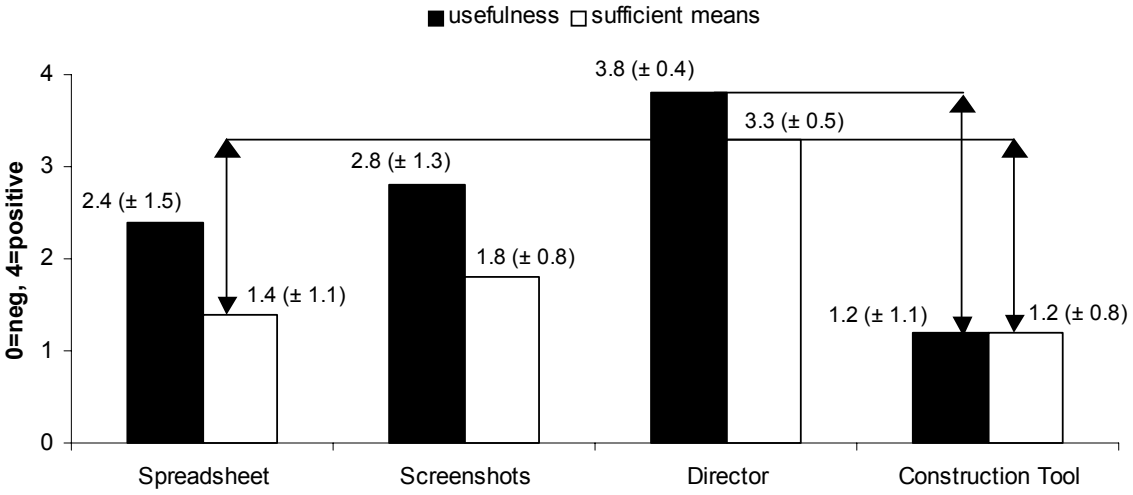

Figure 9: Mean of the 2 statements that discriminated significantly between the four different forms of UI design annotation (N=5; standard deviations between brackets). Arrows designate which items differed significantly in a post-hoc comparison (Scheffé).

The 'Construction Tool' method: The Construction Tool method consists of a rudimentary reproduction of the UI design (made in a freeware SVG drawing program) (Marriott et al., 2002). This reproduction is then annotated with a fixed set of symbols in order to position and size the UI elements (Lok et al., 2001). The symbols can have two colours in order to indicate what their size and position is with respect to fixed screen properties or different screen properties. Fig. 8 shows two examples: how the horizontal and vertical background are positioned and sized (left), and the build up of a vertical menu (right). The annotation is built up using layers that can be (de-)activated in order to show several aspects of the redrawn UI design and their annotations separately. As much layers can be used as needed. UI elements can also be described apart from their context (e.g. labels). This method explains the layout for fixed and different screen properties. Therefore, of all annotation methods, it is most explicit in expressing the design rationale (Bødker et al., 2000) (Moran et al., 1995).

3.2 Experimental setup

Five professional industrial designers highly experienced with interface/interaction applications, who are all closely associated with creating TV-based UI designs, assessed the potential of the four methods of UI design annotations. One designer cooperated already in the first study as well.

After the UI design, the four different forms of annotation methods were presented in the following fixed order: (1) 'Spreadsheet', (2) 'Screenshot', (3) 'Director' and (4) 'Construction Tool'. Each method was discussed for a couple of minutes and evaluated by means of a questionnaire before the next method was introduced.

The evaluation questionnaire (5-point rating scale going from 0='negative' to 4='positive') was identical for all four annotations and asked: "If you, as a designer, would have to produce the <*name of one of the four annotation methods*> example and others would have to use it, how would you rate this with respect to the following…" continued by 12 statements: (1) usefulness (see Fig. 9), (2) easy to produce, (3) added value, (4) applicable, (5) efficient means, (6) effective means, (7) can be produced by me, (8) will be produced by me, (9) improves implementation, (10) reliable, (11) necessary means, (12) sufficient means (see Fig. 9).
After the four evaluations, the designers were asked to make an additional overall summative rating regarding the four annotation methods on a 10-point scale (1='very bad' and 10='excellent'). Next to each question was the option to write free comments.

3.3 Results
To collect qualitative data and to assess the baseline attitude of our interaction designers, we asked how important it is to provide formal additional information to the UI demo design prototypes as input for the software engineering process. They commented the following:
"Starting with design, it would be useful to be able to combine visualization and description and to link that to the software development (version management, application development and so forth…)."
"A formalized version of the design rationale is needed. Detailed visual info can be added to support visualizations."
"Help software engineering to better understand the implementation of UI elements (e.g. slider with annotated info on timing, speed, number of steps)."
"The less ambiguity, the better."
"Some things can be better explained with words, tables, matrixes."
After the four methods were presented, the designers remarked that all the methods more or less suffer from the fact that they are time consuming and prone to errors or misinterpretation. Per method the designers have given the following feedback.
Spreadsheet method: The spreadsheet method is very difficult to produce and to read. The information is distributed too much and misses visual links to the original design. Despite its details, it cannot describe all the diversity of a design and lacks general rules. As a checking tool it can be helpful, but the content should be updated automatically.
Screenshot method: The screenshot method feels familiar to the designers because visual material is reused. They opt for a template though. They indicate that the design rationale can be expressed which makes it easier to deduce commonalities. However, this method will produce a lot of paper and it doesn't force the user to be complete.
Director method: The director method is very complete and can focus on points that need attention. However, it also contains too much different types of information at the same time. It isn't layered well enough ('onion skins') in order to make it usable on several levels. It's questionable whether engineers can benefit. It doesn't show the design rationale because it tackles instances. This method needs to be automated. A toolbox containing the design elements would be helpful.
Construction tool method: This method is very complex (more complex than necessary) and mentally demanding (difficult to learn). This method provides poor information because it is not clear which symbol belongs to which object. It also contains information that is too obvious. They think that it isn't directly related to their deliverables. It might be useful for engineers though.
In their comments designers frequently expressed doubts about the annotation methods because creating them takes extra time and they are prone to errors. Both indicators ('time to use' and 'prone to error') were not able to discriminate significantly between the four annotation methods. However, on the average, per annotation method, almost half of the designers (45% out of a total of N=20 data points, 4 methods * 5 designers) reported the 'time' problem at least once. The same (45% out of a total of N=20) holds for their concerns about 'errors'.
Two of the 12 rating statements of our evaluation questionnaire discriminate significantly between the four annotation methods: 'usefulness' ($F=4.28$, $df=3$, $p=0.021$) and 'sufficient means' ($F=5.94$, $df=3$, $p=0.006$). In addition, an ANOVA was performed using Scheffé post-hoc comparisons, in order to discover which ratings differed significantly. Fig. 9 shows the mean (and standard deviation) of these two significant ratings. A discriminant analysis using only these two items revealed that 70% of original grouped cases per method were correctly classified. The overall summative rating afterwards, did not reach significance among methods.

3.4 Discussion
At begin of our empirical investigations, industrial designers expressed great interest in having formal information additional to the UI design (Fig. 5). This additional information could be in the form of annotations, which the Director method is useful and sufficient at delivering. The fact that the Director method, which was developed by designers (see our study 1 above), is most preferred confirms that user involvement is recommendable (Rauterberg et al., 1992). The Director method is able to limit the

occurrence of inconsistencies and provides information about the design rationale.
However, the design rationale can possibly be expressed more explicitly by means of an additional text document (or audio remarks), especially when deviations to earlier designs need to be justified. Since designers do make use of incremental design, incorporating such additional text documents or audio remarks, might improve the Director method (Mackinlay, 1988) (Mackay et al., 1995). Another drawback of the Director method is that it does not take different screen properties into account.
The Construction Tool method, potentially the most powerful method with respect to different screen properties, is rated surprisingly worst. However, from our first study it became clear that designers so far do not consider different screen properties within their UI design. Our second study confirms that designers only deal with different screen properties by creating a *new* UI design when the properties of the target screen differ. It has to be said though, that the Construction Tool method is difficult to apply, especially with respect to different screen properties. Further, like the Director method, the design rationale is explained mainly by means of additional symbols.
The Spreadsheet method lacks a direct link with the UI design and is a numerical and textual, rather than a graphical method. Since industrial designers are mainly visually orientated, this method does not go along well with the way they act and think. Furthermore, the design rationale cannot be expressed explicitly with this method and therefore it mainly supports consistency as required by the software engineers.
Finally, the Screenshot method, rated second best by the designers, only provides a loose specification regarding the way annotations should be produced. Therefore, it is up to designers to underline the aspects of the UI design that they think are important. Designers probably appreciate the method because it is easy to use (less formal than the other methods) and visually oriented. Besides, the design rationale can be expressed in words as well.
Whether the Director method should be introduced as a standard in the product creation process remains to be seen. The doubts designers expressed about the annotation methods need to be considered as well. Creating written annotations with the Director method takes considerable extra time. Future research has to prove whether audio annotations could overcome this limitation (see e.g. Mackay et al. 1995). However, the time spends on annotations might payoff in the total production process if software engineers can implement the UI more effectively. The possible errors the annotations can contain are not so problematic because software engineers can make use of the UI design in form of the interactive prototype as well. The chance that a UI prototype as well as its annotations contains the same error is quite unlikely.
Producing the annotations turned out to be more difficult than expected, especially when different screen properties need to be incorporated. At present, it seems too early to take different screen properties into account within only one UI design.

4. General Discussion

Two studies are performed in order to bridge the gap between interaction designers of TV-based UI design and software engineers who need to convert this UI design into a complete functional UI. In the first study, it was investigated whether a UI design lends itself for automatic layout and scaling. It turned out that current UI designs can not support automatic layout and scaling because UI elements are not positioned and sized in relative, but in absolute terms. Therefore, software engineers cannot use one UI design in order to convert that into a UI for several products with different screen properties. Nearly every TV product still needs its own UI design and implementation process.
The second study focussed on the clarification of the UI design itself. In our first study the designers had made use of annotations on top of the UI design in order to clarify their design (called Director method). In our second study, the Director method was evaluated by comparing it with three other annotation methods.
The Director method developed by the industrial designers themselves was most preferred, which confirms that involving the user (e.g. designers) is the right approach (Rauterberg et al., 1995). With the Director method a UI design is annotated mainly by producing layout grids and guides around the UI elements in order to shed light on the design rationale. The design rationale can be of use to reduce the occurrence of inconsistencies in the UI design and its implementation. However, the Director method does not take different screen properties into account, unlike the Construction Tool method. This Construction Tool method was the most explicit method regarding different screen properties. However, designers least appreciated it. Therefore, this second study also seems to confirm that fully automatic layout and scaling is yet a bridge too far and needs further research.
Motivated by this investigation, our industrial designers are currently exploring methods to write down the design rationale in order to explain the reasoning behind design decisions. Such a design rational should especially clarify the deviations from earlier UI designs because a UI design is typically based upon earlier versions (incremental design) (Rauterberg, 1996). Therefore, the Director

method might need to be extended in order to clarify such aspects as well. The Director method is not easy to produce, takes considerable time to create, and is by no means complete. Therefore, alternative methods to express the design rationale more easily should be explored (Caroll, 1995) (Bødker et al., 2000). There is some evidence that design and software implementation should remain separate because the creative expertise of interaction designers – creating appealing UIs – can have a positive effect on usability (Kurosu et al., 1995) (Tractinsky, 1997) (Karvonen, 2000) and therefore should not be too constrained by engineering requirements.

When the design and engineering disciplines remain separated, the product creation process can be improved if the understanding of each other's work processes could be increased. For instance, the awareness of software engineers can be increased when the design rationale is explained properly (Moran et al., 1995). Also, designers can be involved more in the product creation process, by showing them intermediate visual results of the implementation. Future improvements of the product creation process should therefore make both disciplines more aware of each other's work work (Rauterberg et al., 1992) (Rauterberg et al., 1995) (Rauterberg, 1996).

5. Conclusions

Within the domain of UI design, research in design support has generally followed different approaches. Our study focuses on uncovering and recording the rationale used to arrive at different UI designs (Moran et al., 1995). The Director Method concentrates on providing a shared UI for expressing designs and reflecting upon how these UIs are used by industrial designers as input for software engineers. Our aim is to draw from the experiences of both these groups. The intent of UI design rationale is documentation of the sequence of decisions made in realising a design.

In our multinational company, the current television-based UI designs do not support automatic layout and scaling. UI designers specify position and size of UI elements in absolute terms. With relative position and size, software engineers, who need to convert the UI design into a fully functional UI, could have been enabled to use one UI design and convert that into several fully functional UIs for products with different screen properties.

In our first study, designers explained the rationale behind the UI design by creating annotations on top of it. In our second study, the potential of this annotation method was assessed by comparing it with three other annotation methods. Designers preferred the annotation method created in the first study. However, this annotation method does not support software engineers to use automatic layout and scaling. An annotation method that does support automatic layout and scaling was rated worst by designers. Therefore, it can be concluded that it is yet too early to apply automatic layout and scaling in the product creation process.

The most preferred Director method has other drawbacks as well. For instance, designers make use of incremental design and the Director method is not able to specify the changes and deviations from earlier UI designs. This most preferred annotation method also takes a considerable time to produce an interactive prototype as well. Therefore, in the future, more appealing and easy to use annotation methods should be considered (Vincente, 2000).

Acknowledgments

We would like to acknowledge the participation of the industrial designers from Philips Design, as well as the valuable input from Kees van Overveld, TU Eindhoven.

References

Atwood M.E., McCain K.W. & Williams J.C. (2002). How does the design community think about design? *Proceedings of the ACM Conference on Designing Interactive Systems DIS'02,* pp. 125-132. ACM Press.

Bødker S., Graves Petersen M. & Nielsen C. (2000). Creativity, cooperation and interactive design. *Proceedings of ACM Conference on Designing Interactive Systems: processes, practices, methods, and techniques DIS'00*, pp. 252-261. ACM Press.

Carroll J.M. (1995). *Scenario-based design: envisioning work and technology in system development*. New York: John Wiley & Sons.

Kamps T. & Reichenberger K. (1994). Automatic layout based on formal semantics. *Proceedings of ACM Workshop on Advanced Visual Interfaces*, pp. 231-233. ACM Press.

Kamsties E., von Knethen A., Philipps J. & Schätz B. (2001). An empirical investigation of the defect detection capabilities of requirements specification languages. *Proceedings of IFIP Sixth CAiSE International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design* (EMMSAD'01).

Karvonen K. (2000). The beauty of simplicity. *Proceedings of the ACM Conference on Universal Usability. CUU'00*, pp. 85-90. ACM Press.

Keränen H. & Plomp J. (2002). Adaptive runtime layout of hierarchical UI components. *Proceedings of NordiCHI'02*, pp. 251-254. ACM Press.

Kurosu M. & Kashimura K. (1995). Apparent usability vs. inherent usability – experimental analysis on the determinants of the apparent usability. P*roceedings of ACM* C*onference CHI'95*, pp. 292-293. ACM Press.

Lok S. & Feiner S. (2001). A Survey of automated layout techniques for information presentations. *Proceedings of the SmartGraphics Symposium,* pp. 61-68. Hawthorne, New York.

Mackay W.E, Pagani D.S., Faber L., Inwood B., Launiainen P., Brenta L., Pouzol V. (1995). Ariel: augmenting paper engineering drawings. P*roceedings of ACM* C*onference CHI'95*, pp. 421-422. ACM Press.

Mackinlay J.D. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, Vol. 5, No. 2, pp. 110-141.

Mackinlay J.D. (1988). Applying a theory of graphical presentation to the graphic design of user interfaces. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, pp. 179-189. ACM Press.

Mackinlay J.D. (1991). Search architectures for the automatic design of graphical presentations. *Proceedings of ACM Conference on Intelligent User Interfaces*. pp. 281-292. ACM Press.

Marriott K., Meyer B. & Tardif L. (2002). Fast and efficient client-side adaptivity for SVG. *Proceedings of the 11th International Conference on the World Wide Web,* pp. 496-507. Honolulu, Hawaii, USA.

Moran T.P., & Carroll J. M. (1995). *Design Rationale*. Mahwah, NJ: Lawrence Erlbaum Associates.

Rauterberg M. (1996). Moderation instead of modelling: some remarks about formal and informal reengineering methods. In: R. Koubek & W. Karwowski (Eds.), *Manufacturing Agility and Hybrid Automation I,* pp. 167-170. Louisville: IEA Press.

Rauterberg M. & Strohm O. (1992). Work Organization and Software Development. *Annual Review of Automatic Programming*, Vol. 16, No. 2, pp. 121-128.

Rauterberg M., Strohm O. & Kirsch C. (1995). Benefits of user-oriented software development based on an iterative cyclic process model for simultaneous engineering. *International Journal of Industrial Ergonomics,* Vol. 16, Nos. 4-6, pp. 391-410.

Tractinsky N. (1997). Aesthetics and apparent usability: empirically assessing cultural and methodological issues. *Proceedings of ACM Conference CHI'97*, pp. 115-122. ACM Press.

Vanderdonckt J. (1994). Automatic generation of a user interface for highly interactive business-oriented applications. *Companion Proceedings of the ACM Conference CHI'94*, pp. 123-124. ACM Press.

von Knethen A., Kamsties E., Reussner R. & Shen B. (1998). A comparative case study with industrial requirements engineering methods. *Proceedings of the 11th International Conference on Software Engineering & its Applications*, December 8-10, Paris 1998.

Vincente K.J. (2000). HCI in the global knowledge-based economy: designing to support worker adaptation. *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 2, pp. 263-280.