PRESENTING TECHNICAL DRAWINGS ON MOBILE HANDHELDS

Georg A. Fuchs, Hans-Jörg Schulz, Heidrun Schumann
University of Rostock
Institute for Computer Sciences
Albert-Einstein-Strasse 21, 18059 Rostock, Germany
{georg.fuchs|hjschulz|schumann}@informatik.uni-rostock.de

Abstract: Due to the spread of mobile handheld devices, new application fields like technical maintenance and mechanical inspections open up to their use. Yet, to fully benefit from their modern technical capabilities, standard interaction techniques need to be revised and new ways of accessing the abundance of technical documentation on the handheld need to be conceived. However, the display of technical illustrations on its tiny screen is still a difficult task. In this paper, we discuss an approach specifically conceived for technical drawings that aims at overcoming this difficulty. To underline its practicability, we also present our software prototype that incorporates this approach by utilizing the technical potentials of the standardized vector graphics format SVG.

INTRODUCTION

A meaningful and comprehensive technical documentation is essential for error-free, smooth operations in many industrial applications. It contains all relevant technical information on a product and is the basis for communication between different branches like design, production, assembly and maintenance, for example during the process of building complex machinery. Aside from part lists, procedures and manuals, the major part of technical documentations comprises drawings and schematics.

Technical drawings fall into one of several categories depending on their intended usage. The main distinction is between production drawings and assembly drawings. The former is used during design and production, whereas the latter is used in assembling and maintenance work. Therefore, production drawings generally focus on single component details like dimensions and surface finish, while assembly drawings depict dependencies between parts and subassemblies like relative position and attachment points. Unlike artist conceptions, technical drawings are subject to defined rules and norms regarding e.g. symbols, perspective, and line styles [ISO02].

Today, paper drawings have become all but extinct. Widespread use of Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) methods mean more and more drawings are available and are used throughout all stages of the production process in electronic form. This has the added benefit that illustrations for specific purposes can be derived from a single parent drawing. In this paper, we specifically focus on illustrations required for maintenance work. Servicing machinery in fixed installations, e.g. in power plants, is necessarily performed on-site, thus the service technician needs to bring the required documentation with him. Consequently, the bulk and weight of (potentially multi-part) printed manuals is rather undesirable.

Because of this, combined with the fact that mobile devices like PDA handhelds are becoming more and more common, there exist several projects that seek to utilize these devices as portable “e-handbooks” or even augmented reality (AR) applications, both for assembly [ART06] and maintenance [NPF+04, FR$^+$06] tasks. One assumption for this paper, however, is the use of commonplace mobile devices instead of special viewing hardware or smart environments required for AR approaches. In practice, this constraint might often be based on availability and cost considerations. This inevitably leads to all the problems and technical limitations that such devices struggle with. We found that most comparable approaches concentrate on a single specific aspect, like pipe flow visualization [NPF+04], rather than addressing these in principle.

Therefore in the next section, these principal challenges of are briefly rehashed. In the two sections following thereafter, we present basic concepts that can be used to present technical drawings efficiently on small devices. These sections discuss concepts and techniques for presenting assembly drawings and circuit schematics, respectively, as typically used during maintenance tasks. In the last section we give some concluding remarks.

PROBLEM ANALYSIS

The constriction on run-of-the-mill mobile devices like PDA has some important ramifications. Aside from data security issues, limitations in storage and processing capability mean it is generally not feasible to employ a CAD-Viewer on the device, i.e. use CAD data directly. Therefore, some set of views suitable for a given task at hand must be pre-selected and converted into static illustrations. Even more importantly, due to the small size of typical handhelds, the available screen space is also very limited. The challenge in integrating even static technical illustrations in such an environment becomes apparent when the resolution of a typical PDA is contrasted to the resolution of a medium-sized circuit schematic:

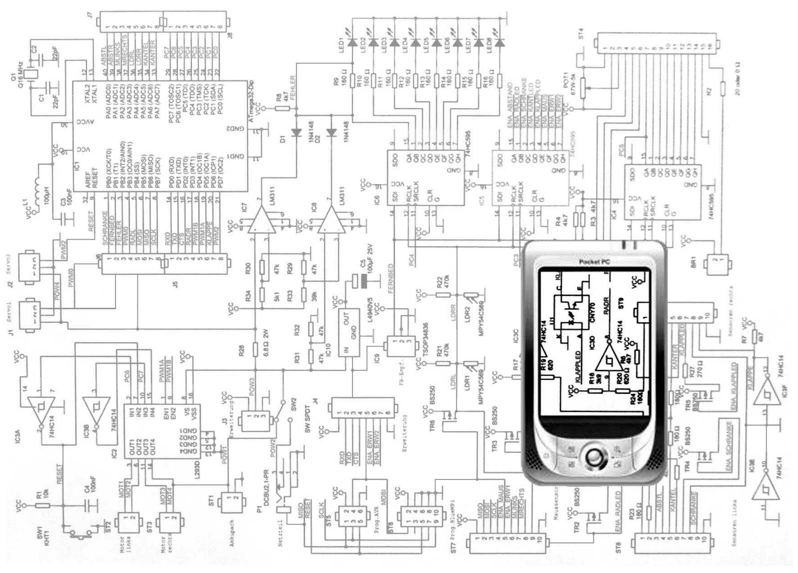


Figure 1: Comparison of the screen resolution of a typical PDA (320x240 pixels) to the resolution of a medium-sized circuit schematic (1742x1275 pixels)

The straightforward solution to the above challenge, scaling the illustration down to the size of the PDA screen, leads to unacceptable results. There would be indiscernible tiny or cluttered details, possibly further distorted by scaling artefacts. Therefore, the adaptation must consider the semantics of the technical drawing in order to select a subset of information to be displayed. This provided, there are basically two ways to solve the screen space problem:

- The display can be limited to show only a certain part of the entire technical drawing that fits the PDA screen, thus sacrificing the completeness of the representation but preserving its detailed depiction. The degree of this limitation can easily be chosen from several levels of segmentation that are appropriate for technical drawings -- like “Complete”, “Functional Modules”, “Sub-Modules” and “Individual Parts”.
- A number of individual objects can be merged into new “compound objects” that take up less screen space, thus sacrificing the in-depth-depiction of all objects but preserving the completeness of the overall structure. This approach can also easily be graded into several levels of detail.

In combination, it is possible to gain results that are optimized with respect to both, completeness and depth, according to the demands of the task at hand that the user needs to carry out upon the drawing.

There are two principal choices for the graphics format used in these approaches: raster images (bitmaps), or vector graphics. The advantage in using bitmaps is that there are numerous established formats and out-of-the-box viewing software available. However, bitmaps have no notion whatsoever about the structure of the depicted content if they are not annotated with external metadata [FR$^+$06]. They also suffer heavily from scaling artifacts, e.g. when originally thin lines are scaled down to sub-pixel width and are thus vanishing.

Vector graphics like the predominant SVG[1] standard, on the other hand, build on graphical primitives like rectangles and circles, and group these to form more complex shapes. Therefore vector formats do exhibit some inherent notion of structure that can be utilized in different levels of segmentation. Moreover, vector graphics are defined by means of geometric coordinates independent of pixel sizes and thus can be arbitrarily scaled without loss of quality. These advantages make vector graphics a much better choice as the basis for technical illustrations. As a matter of fact, many CAD packages provide an option to export selected views as SVG files.

In the following two sections we will discuss ways to use both approaches outlined above efficiently and using SVG drawings, for two typical use cases.

[1] Scalable Vector Graphics, a W3C recommendation for XML-based vector graphics [Jac03].

CONCEPTS FOR PRESENTING ASSEMBLY DRAWINGS

A maintenance technician typically needs assembly drawings at the work site to determine the best sequence of dismantling machinery in question, to locate parts that must accessed or attachment bolts of those to be removed, and detail information on components disassembled for repair. Finally, the technician needs to discern the correct alignment of components upon reassembly. Therefore, the visual representations by a mobile "e-manual" need to address all these different aspects.

Technical drawings for a given unit comprise a hierarchy of single component drawings, subassembly drawings and the general drawing. These drawing types already implement three levels of segmentation as postulated in the previous section. The general drawing will depict a complete overview of the entire machinery (likely with some simplifications), whereas subassembly and component drawings will sacrifice the completeness for subsequently more details locally.

However, these drawings may still require adaptation to fit small PDA screens, and to communicate certain aspects of the task at hand more efficiently. The visibility and perceptibility of relevant objects is essential for intent-based visualizations [SF91]. This is especially an issue when typical step-by-step, automatically generated assembly instructions [AP+03, HP+04] are created from a single (or few) available input images. It must therefore be possible to modify an existing base illustration to:

- Accentuate important parts/components in the illustration, or in reverse, subduing the component's context.
- Hide from the drawing parts irrelevant in the current context, or that occlude relevant geometry. This is especially important since the viewpoint cannot be changed in 2D graphics to resolve occlusions.
- Show details within their context. Very small parts like fastening screws are only drawn in conjunction with larger components, but may become indiscernible small if drawn to scale, especially when considering typical PDA screen diagonals. These parts should be displayed in an enlarged detail view.
- Explode the view to show the relative position and alignment of several parts with respect to each other. Explosion views are also used to show the direction in which parts must be pulled off/put back during assembly [AP+03]. This may require the addition of subsidiary lines for clarity.
- Label the illustration, e.g. for cross-referencing the illustration with additional information, e.g. part lists, or to link labeled view elements with additional audio outputs [FR+06].

In the following paragraphs, concepts to realize the above bullets using SVG are discussed. These concepts have been implemented in a software prototype deployable on Windows Mobile-based PDA.

The SVG format allows to hierarchically group vector primitives with the <g>…</g> tag, to assign a group common display attributes (color, line width etc.), and to assign a reference id to enable reuse of the defined geometry [Jac03]. It also allows embedding content from other SVG files into a composed drawing. Usually, grouping occurs according to graphical criteria, e.g. all background gradients or all lines with the same style. However, this ordering is not mandatory.

Instead, we propose a new concept where the groups are defined according to semantic criteria. First, all vector primitives constituting specific components are grouped together. Since groups can contain subgroups, the logical structure of machinery (single parts grouped into subassemblies grouped into main assemblies) can be reflected in the structure of the SVG file. It is also possible to define several graphical levels of detail for individual components in the same way. Any SVG file, e.g. from a CAD export, can be re-sorted using authoring tool developed for this purpose (Figure 2 (a)). Using these prepared SVG files as input to the "e-manual" application on a PDA, the above features can be supported efficiently.

Accentuation of single components is done by modifying their rendering attributes, e.g. through fill color, line thickness, or explicitly drawing the (usually invisible) bounding box in case of small objects (Figure 2 (b)). Technically, the SVG vector primitives associated with the component group have their corresponding attributes modified before they are passed on to the render engine. Thus, our concept keeps rendering of the content independent of the semantics required to identify the accentuated elements. Our software uses a task model to define the necessary semantics and to control the parameterization of the renderer [FR+06].

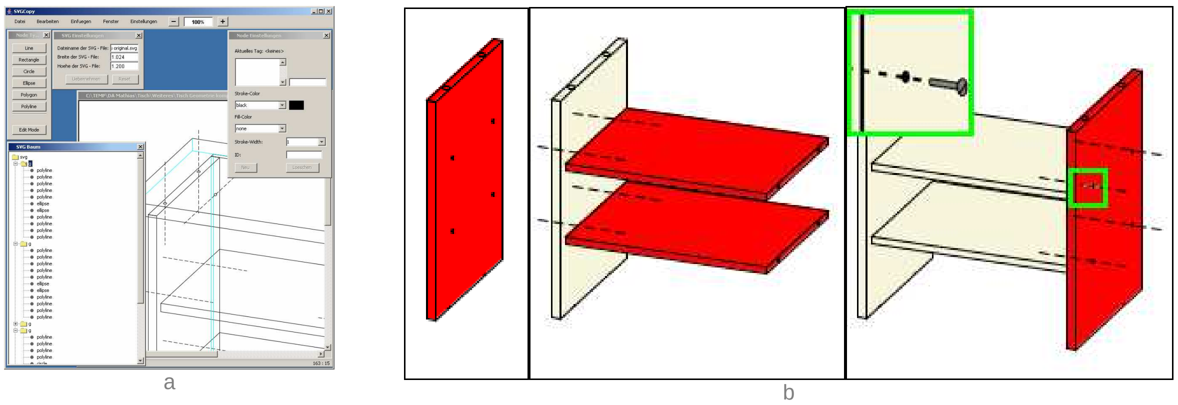
Information hiding can be achieved in a variety of ways. If a component should be removed from the illustration in its entirety, its associated geometry simply is not passed to the renderer at all (Figure 2 (b)). Another approach is to render the component geometry first, before all other components. Since SVG renderer by definition overwrite already drawn elements with later ones, this results the hidden component being pushed into the background. Like accentuation, the required semantics for both approaches are defined in the task model.

Information hiding can also be used as interactive tool [BS+94]. We implemented an “X-Ray lens” that modifies the representation to a wire-frame rendition within the lens area. It is a powerful new concept that allows a user to interactively examine regions of an illustration, revealing geometry otherwise hidden in the default view (Figure 3(a)). It is also usable on any SVG file even if no semantics from a task model are available. Technically, a temporal copy of all primitives within the lens area is created with the fill color attribute removed from the render attribute list. Note that if a primitive is only partially covered by the lens, it is split into two or more temporal primitives as needed to confine the modification to the lens area. The modified primitives are then substituted for the original ones when passed to the renderer.

Detail-in-overview displays can be implemented using SVG’s viewport mechanism [Jac03]. For this, a rectangular region is defined around the part of interest. A viewport with appropriate scaling is defined and positioned on the display area. Then, all geometry within the defined region of interest is rendered into the viewport. The necessary scaling and clipping is done automatically by the SVG renderer (Figure 2 (b), right).

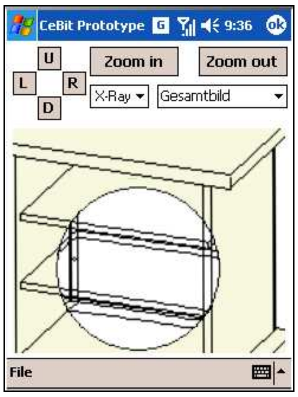
Explosion views can be created by displacing adjacent components so that these do no longer overlap. An automatic approach proposed in [AP+03] determines the displacement direction that affords the least distance by evaluating the bounding boxes of all affected components. However, to allow more complex displacements, in our prototype displacement vectors and subsidiary lines are defined explicitly. This information is stored to the SVG file using the XML namespace mechanism, enabling standard SVG agents to ignore it and still display the unexploded view correctly. Our prototype, on the other hand, uses an extension-aware SVG agent that decodes this information. Figure 3 (b) shows an example of an assembly instruction encoded as a single explosion view.

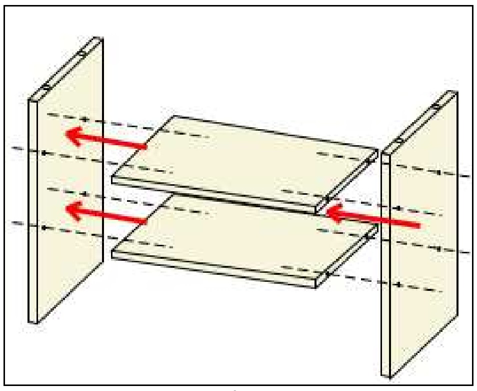
Labeling is an important aspect of technical illustrations [Tuf97], although the labeling problem has proven to be NP-complete [MS97]. Labeling in our prototype is therefore either static, i.e. done by the content author or, if a WLAN connection is available, using a labeling service as described in [FL+06]. In both cases, textual labels are directly supported by SVG, so no extension of the renderer is needed.



a b

Figure 2: Authoring tool for editing SVG group/symbol tag structures of input files (a), example for a step-by-step assembly instruction generated from a single input image using accentuation, information hiding, and detail-in-overview (b).





a b

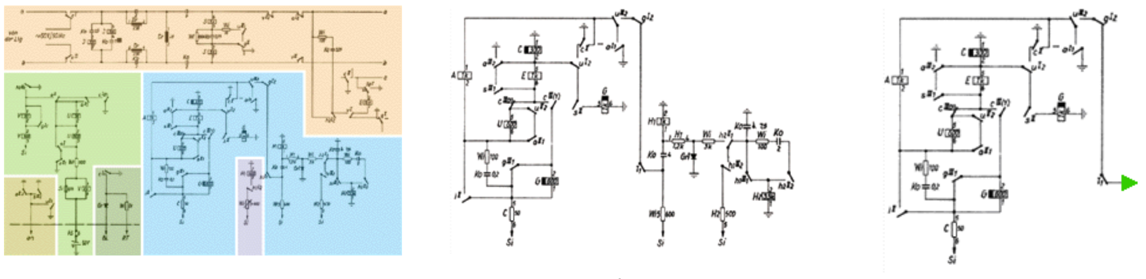
Figure 3: Screenshot of the "X-Ray lens" used to reveal hidden geometry (a), generated explosion view with additional geometry (subsidiary lines) shown (b).

Concepts for Presenting Structures

In contrast to assembly drawings, structure representations like piping diagrams or technical process flows usually focus upon displaying the functional dependencies between different parts or modules. While the individual parts themselves can be represented by one of the techniques described in the above section, the depiction of their relationship to one another adds to the difficulty of bringing such structure representations to the limited screen of a handheld device. Approaches to overcome (or at least lessen) this problem, can again be derived from the two principal strategies – segmenting the structure or decreasing their depth. This section discusses these approaches specifically for circuit schematics in the context of electrical engineering, but the underlying principles can easily be adapted for structures from other technical fields as well.

Segmentation can at first be used to partition the circuit schematic into its disjoint parts. These can be extracted and then displayed independently (see Figure 4 (a) and (b)) or combined into a new schematic as needed. This allows to reduce the content of the illustration to only those segments that are ultimately required for the task at hand. When there are no disjoint parts or further simplification of an extracted segment is needed, a folding method can be employed that hides additional subsections of the circuit drawing (exemplified in Figure 4 (b) and (c)).

Depth reduction uses on the other hand means of abstracting certain parts of a complex schematic. This is usually done by repeatedly combining several low-level parts into one high-level "compound part" that represents the merged elements as a functional block. This is common practice in electrical engineering where functional groups of parts are integrated into a single IC or circuit board to ease its repair and its interchangeability with updated, newer hardware.



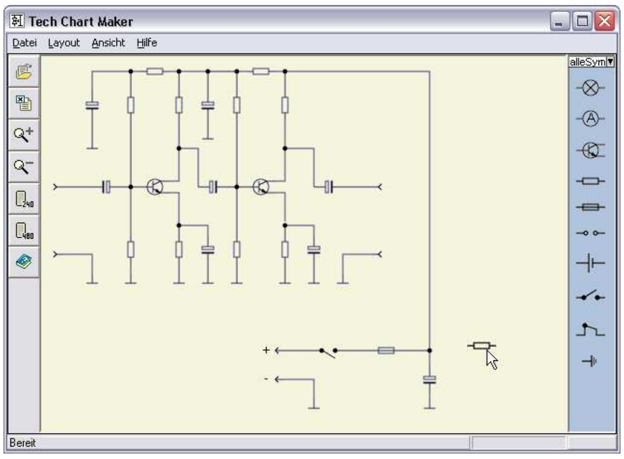
a b c

Figure 4: A circuit schematic that is segmented into its disjoint sections (a), an already larger view of the light blue section only (b) and the even larger depiction of the same light blue section with one hidden part at the green arrow head (c).

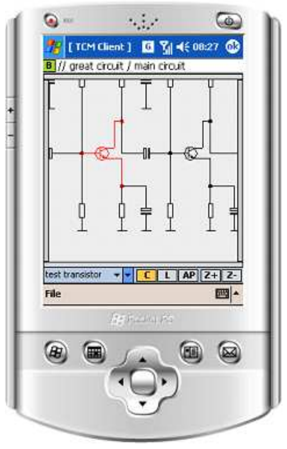
Both of the presented methods suit the typical tasks that an electrical engineer needs to carry out upon a schematic very well. Thus, the aggregation of "low-level" components to reduce the depth of the schematic serves well in an overview that shows the main structure of the entire circuit. And segmentation and folding of the circuit are good ways to focus upon a certain section of the schematic. This is very helpful when measuring sections need to be identified or functional modules and individual parts must be located.

While the above ideas for reducing the visual complexity of circuit diagrams seem quite elementary and uncomplicated at first, their technical implementation is not. Especially the automated segmentation and abstraction of a schematic representation pose algorithmically complex tasks. Their realization is based upon a semantic annotation of the SVG geometry that basically models the topology of the described circuit. Unfortunately, our graphics format of choice – SVG – is only of limited use for their implementation. Since a standard SVG-file [Jac03] does not hold any information about the topology of the structure it contains, e.g. which geometric primitives are connected with each other by which (poly-)lines, an automatic processing of the above methods would be impossible. Yet, there have been first attempts to provide SVG with additional topological features that allow to exactly grasp the notion of geometric structures that are connected to one another [TMM00, KMS04]. While these non-standard extensions will be simply ignored by a standard SVG-viewer, a customized SVG-viewer can take advantage of the additional topology information in the SVG-files and use it for the automatic deduction of possible folding points or unconnected circuits. Furthermore, the topology information also permits to run a final compaction step, for example to further compress the image or to adjust the image to a certain aspect ratio of the screen. Since the compaction of a given two-dimensional layout is NP-complete [SP82], heuristics or approximation algorithms may need to be employed to run the compaction in acceptable time [SL89, Wol84].

If, for certain reasons, one must not extend the SVG standard, an authoring tool can be used to manually annotate the schematic in SVG format with the needed information. Both, the segmentation into its connected parts or the selection of meaningful folding points, can be modeled directly in SVG via the <g>…</g> tag described in the previous section. Since the other parts of the proposed electronic handbook besides the circuit schematics need to be authored too (written text, photographs, video-tutorials, etc.), such an authoring tool for annotating circuit schematics can easily be plugged into the existing workflow and its toolset. While this requires an additional amount of effort on the side of the content provider, it guarantees the compatibility to all standard SVG-viewers. An example for both, our authoring tool and our client software using a standard SVG-viewing component can be seen in Figure 5.



a



b

Figure 5: Our authoring tool for creating and annotating circuit schematics in SVG (a) and the accompanying client software displaying an example circuit with a highlighted transistor using the standard conform SVG-component SVG.Net (b).

Independently of the way in which such a screen-optimized circuit schematic is generated (manually with an authoring tool or automatically from a topology description), the above presented methods usually behave quite well for ordinary schematics. Yet, one has to bear in mind that if one tries to handle extremely large and highly connected circuit diagrams, a suitable reduction of its visual complexity might be hard or even impossible to obtain. Fortunately, since most of the complex functional blocks of a circuit are already integrated into higher-level parts like ICs or circuit boards, our experiences show that overly complex schematics occur only rarely in real world maintenance applications.

SUMMARY

As one can easily see from the above sections, providing CAD-derived illustrations for mobile handheld devices is an ambitious goal that gets increasingly difficult with the amount of automation involved. Especially the reduction of complex drawings to fit the small screen space of today's mobile devices poses a challenging problem. Yet, it is encouraging to see that already the basic techniques presented in this paper combined with the technical possibilities of a state-of-the-art vector graphics format like SVG show promising results. Also, the possibility to shift complexity issues and issues of standard conformity between the authoring process and the interactive mobile client allows the adaptation to a wide range of technical architectures and content creation workflows. In this paper, we have shown both – powerful authoring tools and prototypes of smart mobile clients that utilize their respective strategies to circumvent technical incompatibilities as well as runtime and screen bottlenecks.

Our ideas to further this study focus on additional automation possibilities using topologically annotated illustrations. Because describing the topology using a graph structure seems to lead towards unexpected benefits like the possibility to use a simple breadth-first-search to easily determine connected components or graph drawing algorithms for further optimization of the layout. Additionally, an evaluation of the proposed techniques and their acceptance by maintenance workers in different scenarios needs to be realized.

References

[AP+03] M. Agrawala; D. Phan; J. Heiser; J. Haymaker; J. Klingner: Designing Effective Step-By-Step Assembly Instructions; Proc. of ACM SIGGRAPH 2003, S. 828-837

[ART06] ARTESAS. *Advanced Augmented Reality Technologies for Industrial Service Applications*. http://www.artesas.de, 2006.

[BS+94] E. A. Bier, M. C. Stone, T. Baudel, W. Buxton, K. Fishkin: *A Taxonomy of See-Through Tools*. ACM Proceedings of CHI '94 (Boston, MA, April 24-28), S. 358-364, 1994.

[FL+06] G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, H. Schumann, Th. Strothotte: *Adaptive Labeling for Interactive Mobile Information Systems*, 2006.

[FR+06] G. Fuchs, D. Reichart, H. Schumann, P. Forbrig. *Maintenance Support - Case Study for a Multimodal Mobile User Interface*. IS&T/SPIE's 16th Annual Symposium Electronic Imaging, Jan. 2006.

[HP+04] J. Heiser, D. Phan, M. Agrawala, B. Tversky, P. Hanrahan: Identification and Validation of Cognitive Design Principles for Automated Generation of Assembly Instructions; Advanced Visual Interfaces 2004, S. 311-319

[ISO02] ISO Standardization Committee. *ISO Standards Handbook Vol.1+2*, 4th Edition, 2002.

[Jac03] D. Jackson (Editor). *Scalable Vector Graphics (SVG) 1.1 Specification*. W3C Recommendation, 14 January 2003.

[KMS04] M. Klima, P. Halabala, P. Slavik. *Semantic information visualization*. CODATA Workshop Prague, 2004.

[MS91] J. Marks and S. Shieber. *The Computational Complexity of Cartographic Label Placement*. Technical Report TR-05-91, Center for Research in Computing Technology, Harvard University, 1991.

[NPF+04] W. Narzt, G. Pomberger, A. Ferscha, D. Kolb, R. Müller, J. Wieghardt, R. Bidner, H. Hörtner, C. Lindinger. *PARIS - Personal Augmented Reality Information System.* 2nd ACM International Conference on Mobile Systems, Applications and Services, June 2004.

[SF91] D. Seligmann, S. Feiner. *Automated generation of intent-based 3D Illustrations*. Proceedings of ACM SIGGRAPH, p. 123-132, 1991.

[SL89] H. Shin, C.-Y. Lo. *An Efficient Two-Dimensional Layout Compaction Algorithm*. Proceedings of the 26th ACM/IEEE Conference on Design Automation, p.290-295, 1989.

[SP82] S. Sastry, A. Parker. *The Complexity of Two-Dimensional Compaction of VLSI Layout*. Proceedings of ICCC-82, p.402-406, Sept. 1982.

[TMM00] J.J. Tirtowidjojo, K. Marriott, B. Meyer. *Extending SVG with Constraints*. In Proceedings of the 6th Australian World Wide Web Conference, 2000.

[Tuf97] E. R. Tufte. *Visual Explanations: Images and Quantitatives, Evidence and Narrative*. Graphics Press, Cheshire, Connecticut, 1997.

[Wol84] W. Wolf. Two-Dimensional Compaction Strategies, PhD Thesis, Department of Electrical Engineering, Stanford University, 1984.