Technical Documentation of Software and Hardware in Embedded Systems

Beate Muranko Rolf Drechsler
Institute of Computer Science
University of Bremen
28359 Bremen, Germany
Email: {bmuranko, drechsle}@informatik.uni-bremen.de

Abstract— **Embedded systems are characterized by the presence of software and hardware components. They are integrated e.g. into telecommunication or products such as cars. Due to the size of embedded systems and the reuse of components, documentation of them becomes more important. Although the importance of documentation has increased over the years, it is still a largely neglected part of the development process.**

In this paper we discuss the integration of the technical documentation in the software and hardware development processes. Therefore, we analyze and evaluate classical software and hardware development models with regard to technical documentation. Furthermore, we present a workflow for documentation which is derived from practical experience and can be integrated in existing software and hardware development models. As a proof of concept we present an approach for the integration of documentation techniques into one software development workflow.

I. Introduction

Embedded systems are integrated e.g. into telecommunication or products such as cars. Embedded systems can be analyzed from different perspectives e.g. by looking at the number of processors. According to [1] "about 79% of all the processors are used in embedded systems". Furthermore, there it is reported that "for many products in the area of consumer electronics the amount of code is doubling every two years". Usually embedded systems involve hardware and software, for this both has to be taken into consideration. There are standard components which can be reused, hence the design process does not necessarily start from scratch.

Due to the size of embedded systems and the reuse of components, documentation of software and hardware strongly supports the understandability and use of them. Therefore, as the first step of a research study our interest was in the status quo of present technical documentations. An analysis of the market revealed the following:

On the one hand technical documentation is widely regarded as of little scientific value. Therefore, very little has been published on this subject. Furthermore, software and hardware development, respectively, are considered as much more important than the technical documentation of the developed system. This for instance becomes obvious in [2], where documentation is mentioned only briefly.

On the other hand existing documentation usually does not match the current version of the project, is incomplete and often too complicated. However, technical documentation is a substantial component of the overall product and should be given the same attention as the design and implementation. In more than one case, insufficient documentation has been identified as a source for design error, see e.g. [3].

Beside this, our study showed that there are not many research papers in the area of documentation:

There are tools for automatic source code documentation: An example is the "Javadoc" tool[1] provided by SUN Microsystems, which generates an API documentation in HTML starting from comments in the source code. Furthermore, there is the "CppDoc" tool[2] which also generates HTML documentation. This is done by extracting the source code and special comments from C++ classes. The output of CppDoc and Javadoc are similar. These tools are helpful for providing an initial source code documentation, but they do not give a systematic way of deriving a complete documentation.

Additionally there are initiatives for documentation. An example is the SPIRIT Consortium initiative[3]. The SPIRIT Consortium Specification focused on improving and establishing the automated integration of IP into design flows. This is done by creating specifications in two areas: "an IP meta-data description specification and an interface for integrating IP generators and point-tools" [4]. Thus, the focus is on IP integration (and by this also covers some documentation aspects), but not the documentation flow itself.

Our approach considers the problem from a different perspective. We study new aspects in terms of creation of manuals and other important documents.

Hence, in this paper we examine the status quo of the integration of documentation into software and hardware development processes. Afterwards we take a closer look at classical development models, which we analyze with respect to the way they address documentation issues. Then, we present a workflow for documentation which is derived from

[1]http://java.sun.com/j2se/javadoc/
[2]http://www.cppdoc.com/
[3]http://www.spiritconsortium.com/

practical experience. Based on this we show an integration of documentation into a software model.

This paper is structured as follows: Section II gives a short introduction to the theoretical background of technical documentation. In Section III we give a brief overview of software and hardware development models and analyze them with emphasis on documentation. Then we analyze each model individually with respect to the integration of documentation techniques. The results of this analysis are illustrated and discussed. Section IV presents a prototypical workflow of a general documentation process. In Section V we present an approach for a general solution for the integration problem. Finally, Section VI discusses the results of this paper and a direction for future research is given.

II. Background

This section provides the theoretical framework that is necessary for the understanding of the paper. The terms "technical documentation" and "internal/external documentation" are defined and a listing of possible documentation types is given. Finally, we give a brief overview of German DIN EN 62079, which is the widely accepted standard for the creation of technical documentations in Germany[4].

- **Technical Documentation**
  According to [5], the technical documentation covers all necessary information related to the product and its use. This is recorded either on paper or on electronic media. Furthermore, it is required that all information from the very beginning of the project scheduling is collected and stored. This continues during the entire product lifetime.
- **Internal/External Documentation**
  The term of internal documentation comprises all instructions and information which are meant exclusively for internal use by staff members. Examples for this are product requirement specifications, construction and production documents and quality assurance documents. External documentation includes all product-related customer information and instructions which are delivered to the customer along with the product, i.e. the so-called user information. Examples for this are manuals and operating instructions (see e.g. [6]).
- **Documentation Areas**
  According to [7] documentation can be subdivided into four areas: project documentation, development documentation, product documentation and user documentation. Technical documentation encompasses development documentation, product documentation and user documentation (see Figure 1), all of which are briefly described in the following. The project documentation will not be considered in the following as it is not part of the technical documentation.
  - Development Documentation
    This part of the documentation is concerned with describing problems and their solutions and workarounds, respectively, as well as any additional utilities that have been used for the above-mentioned solutions.
  - Product Documentation
    This may contains e.g. an overview of the product as well as a more detailed description, which includes the program sequence and the underlying data model.
  - User Documentation
    It deals with instructions for operating the product, e.g. in terms of user manuals, maintenance instructions, training course documents and sales documentation.
- DIN EN 62079
  In Germany for reasons of liability, producers, manufacturers and importers of technical systems are required by law (e.g. product liability law) and various regulations (VDE regulations, standards) to supply documentation for their products. DIN EN 62079 contains a general basis for the outline and creation of technical documentation. The application range of the DIN EN 62079 standard covers topics from instructions for small and simple products (e.g. a jar of paint) to large and highly complex products (e.g. industrial facilities) [8]. According to [8] no universally valid standard can cover each individual case that might arise. For this reason additional standards pertaining specifically to the product in question must be consulted previously to the creation of documentation for that product. The DIN EN 62079 standard does not provide any detailed information regarding specific product ranges like software or hardware. Therefore, there is no specialized basis for technical documentation in these areas, like e.g. embedded systems.

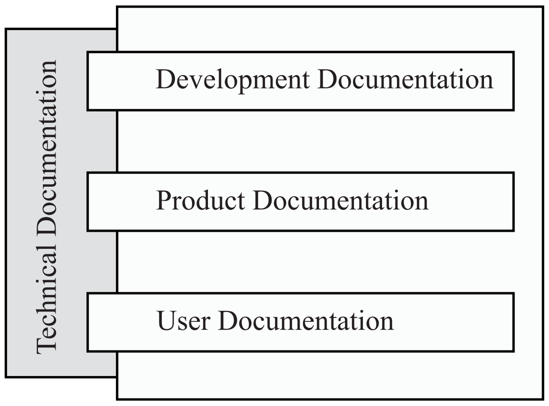

Fig. 1. Documentation areas

[4]In the following we refer to the standard German textbooks on documentation where needed. These books are given as further readings, but are not essential for the understanding of the paper. Similar books can be found in different languages as well.

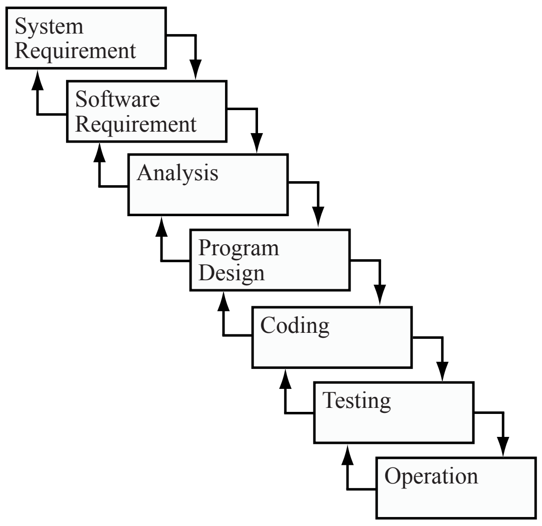

Fig. 2. Waterfall model

Additionally, the DIN EN 62079 standard also contains a check list for analyzing instruction manuals. If used correctly, this check list allows the evaluation and error testing of manuals on a structural level. However, it does not help with content-specific details related to hardware or software.

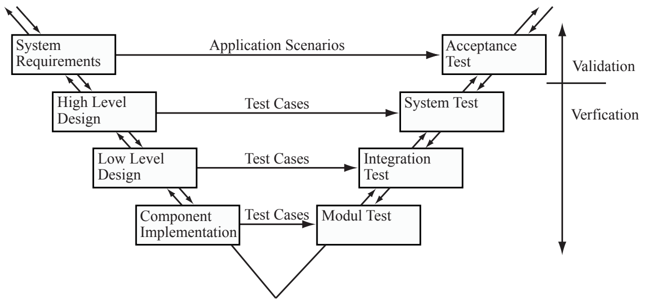

Fig. 3. V-model

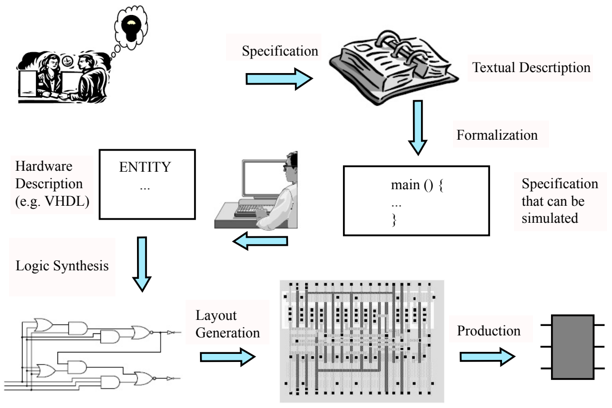

Fig. 4. Hardware model

III. Software and Hardware Development Models

The insufficient attention given to technical documentation in software and hardware development models provide the background for the reevaluation of these models. The goal of the analysis in this section is to point out the above-mentioned lack of consideration for documentation even in such models which explicitly contain the production of documentation as an integral component of their procedure. With regard to hardware development, we discuss the generally used development model which is described in [9]. Specifically, we present the procedure given in [10] which is based on [9] and discusses the classical workflow of development in phases.

According to [11] the Waterfall model [12] and the V-model [13] belong to the class of documentation oriented development models. In the following the Waterfall model and the V-model given in [11] which is based on [13], are briefly described.

The Waterfall model describes the production of software as a sequential process including incremental development phases (see Figure 2). The primary advantage of this model is that it requires a minimal management effort for the development of a product [11]. This is achieved by structuring the development in such a way that one phase must be completed before the next can start. Since the model is document-driven, each phase results in documents, which then are carried along into the following phase. Therefore, it is characteristic for the Waterfall model that receiving and responding to feedback is only possible in between the actual production phases.

The primary goal of the V-model is to integrate a quality assurance into the development process (see Figure 3). In order to achieve this, the V-model includes verification and validation concepts. In this context, verification is defined as the examination of the correctness of the system, i.e. a system is correct if it matches its specification. Similarly, validation ensures that the result of the development process is a system which deals in a suitable manner with the problem it was meant to solve. Since both validation and verification are non-discrete processes, which cover the whole duration of a project's lifetime, their requirements result in a break-up of the strict temporal flow of phases as required by the Waterfall model. Although the V-model is no longer hierarchical, its phases still result in documents.

The established procedure in hardware development [9], [10] is slightly different from the described software development (see Figure 4). For hardware, the development process starts with a textual specification, which is then formalized and described in a programming language that allows simulation. Typically, this is done in C or C++. Starting from this description the coding of the RTL model in a hardware description language, like Verilog or VHDL, starts. The resulting code is then translated automatically into a net list. The net list

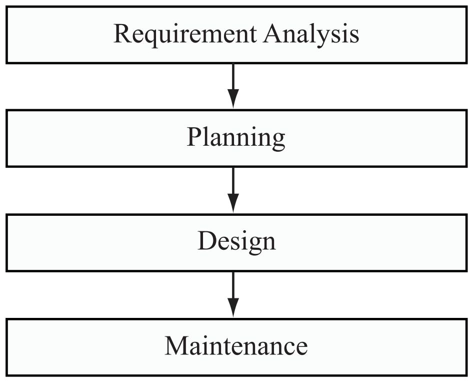

Fig. 5. Documentation workflow

represents the starting point for the production of the chip.

This short summary of the development models already shows that while the individual phases and their connections are clearly defined, the technical documentation is neglected.

When we consider the software development models we have discussed previously, it becomes obvious that while both models mention the need for documentation, neither of them explicitly indicates in which section of which phase the technical documentation is to be created.

The classical hardware development model deals more drastically with the technical documentation by "ignoring" it. Also in verification orientated development models the documentation is not explicitly addressed [14].

By this, we draw the conclusion that in general, the existing software and hardware development models address technical documentation only insufficiently.

IV. Documentation Workflow

In this section we construct a general documentation workflow based on empirical experience [15], [16], [17]. The process is divided into four phases as shown in Figure 5. These are characterized as follows.

- **Requirement Analysis**
  Here, the project has to be analyzed with regard to any requirements of the documentation. Then the target group for any instruction material needs to be determined in order to establish the level of detail necessary for the material in question. If this group has little previous knowledge in handling the product, the documentation needs to be detailed accordingly.
  As soon as these general parameters have been established, the author of the documentation needs to gain detailed knowledge of the product. This leads to a first version of the product description which has to be coordinated with all persons involved in the project, in order to promptly determine and eliminate misunderstandings. As a final step in this phase it is important to calculate a complexity estimate, e.g. the number of pages or the overall cost.
- **Planning**
  In the planning phase, responsibilities are divided among the participants of the project. Additionally, the layout, the internal basic structure and the level of detail of the documentation are precisely specified. This is also the phase where the product itself is examined with regard to compliance with any specific legal standards. For example, it must be established whether the product respects current safety standards, or whether its life cycle matches any minimum warranty periods as prescribed by law. The result of this phase is a complete structure and a temporal operational sequence of the documentation.
- **Design**
  The production phase consists of three documentation releases: the alpha, the beta and the final version. The respective versions of the documentation are created in parallel to the corresponding implementation phase with the assembly of the final product version. The alpha version already contains descriptions for the entire functionality of the product and all its components, in accordance with the previously established documentation plan. In this phase all tools which will be employed in the creation of the final documentation should also be tested, in order to ascertain that they meet the requirements, too.
  Once completed, the alpha version is verified by the staff members responsible for the respective project parts. Any comments, revision suggestions and improvements are included, resulting in the next version. The beta version will once again be presented to developers for repeated examination. Additionally, in this release an evaluation by external experts and end users should be carried out. The result of this combined revision process is the final documentation version.
- **Maintenance**
  No matter how carefully a manual is put together, some weaknesses and errors will always remain. These tend to emerge gradually when the manual is widely used over a predetermined period of time, and they cannot simply be ignored. It becomes clear that, far from being static, documentation is a continuous product which needs to be revised constantly.

V. Integrated Approach

We have shown in the previous sections that the lack of accepted documentation procedures in development models is a problem which needs to be solved. Our approach for a solution integrates the documentation workflow we illustrated in the previous section into the Waterfall model. In Figure 6 the resulting model is illustrated by lining up the original model and our documentation extension and establishing the connections between their respective phases.

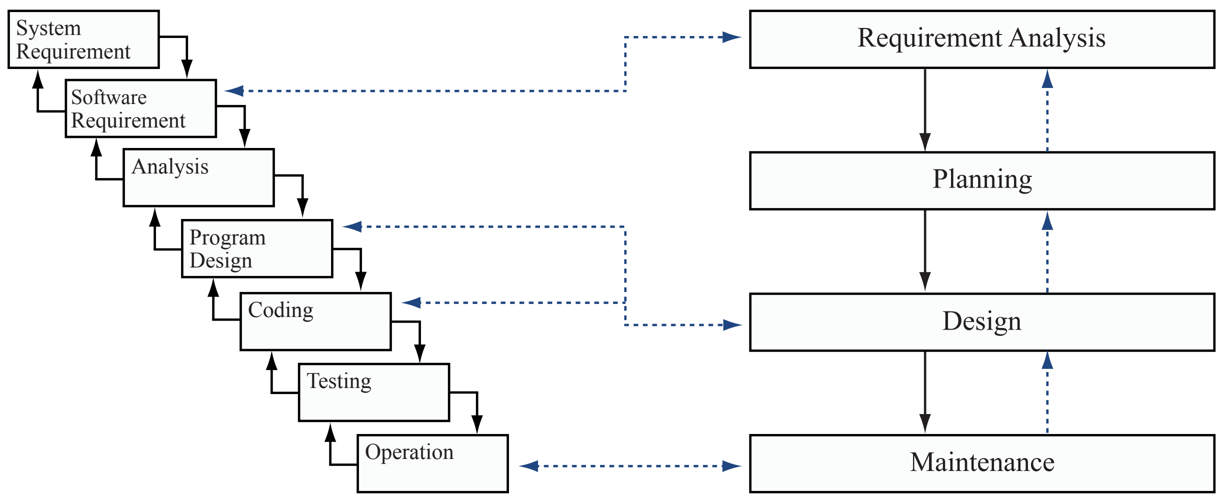

Fig. 6. Initial model

On the right hand side of the illustration the workflow for the documentation is displayed, which is extended by feedback connections between the individual phases. These repeated opportunities for feedback ensure that sufficient interaction between phases can take place. They are represented by dashed arrows connecting each phase with its predecessor. An example for feedback is the possibility to go back from "maintenance phase" to "design phase". On the left hand side, the Waterfall model is displayed.

The compound model shows explicitly where documentation needs to be integrated into the waterfall model. This integration is symbolized by dashed arrows which connect both workflows.

In the following description of our compound model we only refer to those phases which have been extended by additional documentation steps.

- **Software Requirement**
  As the software requirements are established, it is crucial that the general parameters for the documentation are established with them. It is also necessary to determine which target groups the manual will address. Consequently an estimate for the level of detail required has to be identified. This also holds for any subsequent descriptions and manuals. Additionally, at this point the type of the documentation must be determined, e.g. book, compact disk, web format.
- **Program Design and Coding**
  At the design phase, the software is modeled. This provides an overview of the system's architecture and components. In the coding phase the designed model is implemented. Documentation is integrated into both phases, because the versions of the documentation (alpha, beta and final) have to be composed simultaneously with the product. As a result of the integration we derive a

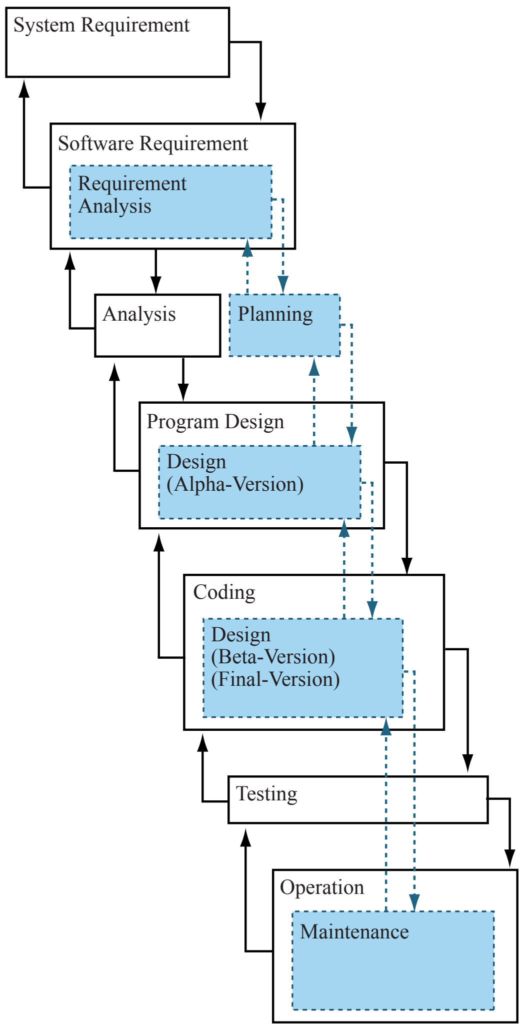

Fig. 7. Merged model

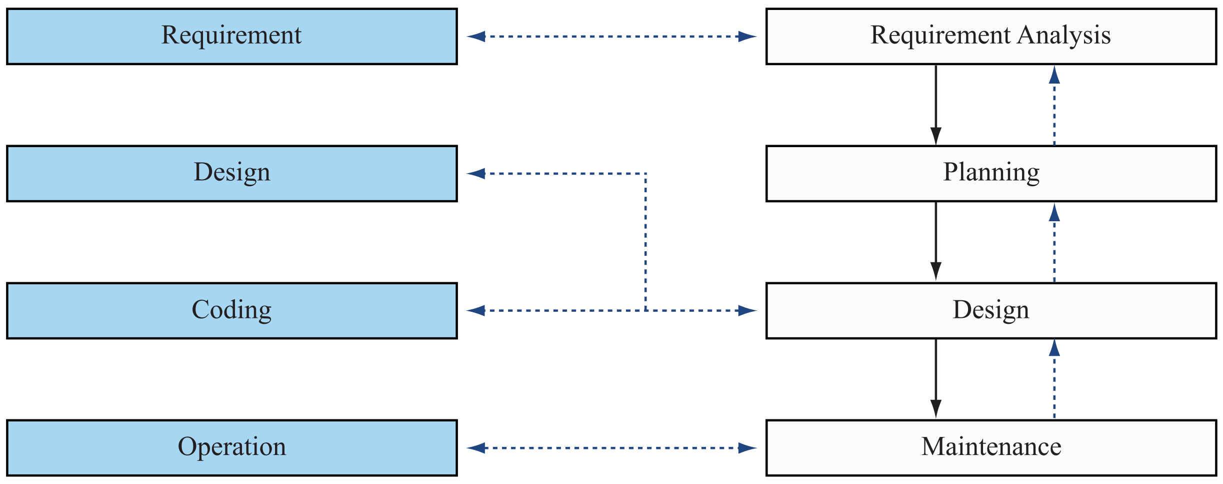

Fig. 8. Universal model

synergy between the processes of production and documentation. Updates in the design will be directly echoed in the documentation, so that the documentation always reflects the status quo.

- **Operation**
  Some errors will only be found when the product is operated. As soon as they are fixed in the software, the documentation will also have to be altered.

Our approach of the overall compound model can be seen in Figure 7. The documentation steps have been merged into the Waterfall model for a better view of how the new development process is structured.

On the basis of the combined model it is obvious that not all phases of the development workflow are supplemented by documentation. Therefore, Figure 8 displays those phases which explicitly include documentation steps, disregarding the others.

VI. Conclusion

In this paper we evaluated the integration of documentation into software and hardware models that form the process for embedded system development. In order to deal with this, we presented the necessary background information. This part ranged from on overview of the individual areas of documentation to the DIN EN 62079 standard to an analysis of the currently established software and hardware development models. This analysis resulted in the conclusion that technical documentation for software and hardware is treated superficially and insufficiently in the DIN EN 62079 standard as well as in the development models we evaluated.

We described a documentation flow and discussed it in the context of the Waterfall model. An integration has been presented. By this, we gave a framework for a documentation flow for embedded system design.

The focus of the current work is to extend the approach to other models, like the V-model discussed above.

References

[1] P. Marwedel, *Embedded System Design*. Kluwer, 2003.
[2] M. Keating and P. Bricaud, *Reuse Methodology Manual: For System-on-a-Chip Design*. Kluwer, 2002.
[3] B. Bentley, "Validating the Intel Pentium 4 microprocessor," in *Design Automation Conf.*, 2001, pp. 244–248.
[4] C. Lennard, "Industrially Proving the SPIRIT Consortium Specifications for Design Chain Integration," in *DATE Design Automation and Test in Europe*, 2006, pp. 142–147.
[5] C. Krings, *Wissenschaftliche Grundlagen der Technischen Kommunikation*. Gunter Narr, 1996.
[6] G. Pötter, *Die Anleitung zur Anleitung: Leitfaden zur Erstellung technischer Dokumentationen*. Vogel, 1994.
[7] H. Scheibl, *Wie dokumentiere ich ein DV-Projekt?: Dokumentationsverfahren in Theorie und Praxis*. expert, 1985.
[8] DIN EN 62079, *Erstellen von Anleitungen: Gliederung, Inhalt und Darstellung*. VDE, 2000.
[9] G. DeMicheli, *Synthesis and optimization of digital circuits*. McGraw-Hill, 1994.
[10] R. Drechsler and A. Breiter, "Hardware project management - what we can learn from the software development process for hardware design," in *4th Conference of Informatics and Information Technologies*, 2003.
[11] H. Balzert, *Lehrbuch der Softwaretechnik: Software-Management Software-Qualitätssicherung Unternehmensmodellierung*. Spektrum, 1998.
[12] B. Boehm, *Software-Engineering Economics*. Englewood Cliffs: Prentice Hall, 1981.
[13] IABG Information Technology, *V-Model: Lifecycle Process Model*. www.v-modell.iabg.de/kurzb/vm/k_vm_e.doc, 1993.
[14] L. Bening, *Principles of Verifiable RTL design*. Kluwer, 2002.
[15] T. Barker, *Writing Software Documentation: A Task-Oriented Approach*. Longman, 2003.
[16] A. Sikora and R. Drechsler, *Software-Engineering und Hardware-Design: Eine systematische Einführung*. Hanser, 2002.
[17] J. Price and H. Korman, *How to Communicate Technical Information: A Handbook of Software and Hardware Documentation*. Addison-Wesley, 1993.