Institute of Software Technology
Department Software Engineering
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelor Thesis No. 109

Documentation of Modules of a Production Line Software

Andreas Gross

Course of Study: Software Engineering

Examiner: Prof. Dr. rer. nat. Stefan Wagner
Supervisor: Ivan Bogicevic, Dipl.-Inf.
Commenced: 2014-01-01
Completed: 2014-07-02

CR-Classification: D.2.7

Contents

List of Figures

List of Abbreviations

ERP	Enterprise resource planning
Fmst	Conveyor control (Ger. Fördermittelsteuerung)
IEEE	Institute of Electrical and Electronics Engineers
ISTE	Institute of Software Technology
JVM	Java Virtual Machine
LBMS	Line Browser Module Server
PDF	Portable Document Format
Perf	Performer
PLC	Programmable Logic Controller
SVN	Apache Subversion
UML	Unified Modeling Language
UniMoDoc	Unified Module Documenter
XML	Extensible Markup Language

Abstract

This thesis investigates the documentation of a Production Line Software for type, quality and structure. A documentation scheme for the Production Line Software is created with the documentation scheme of the software documentation tool UniMoDoc. The created documentation scheme is used for a post documentation for a selected part of the Production Line Software. A case study is processed with the post documented UniMoDoc document which is verified for the utility of the documentation scheme. Additional interviews to the case study allow an evaluation of the documentation scheme, which lead to the result of this thesis. This thesis aims an abstract documentation scheme to understand the overlapping functionalities of software modules. An abstract documentation procedure is used in this thesis instead of a pure source code documentation procedure.

Zusammenfassung

Die Bachelorarbeit untersucht Dokumentationen einer Produktionslinien-Software auf Art, Qualität und Struktur. Gemeinsam mit dem Dokumentationsschema des Dokumentationswerkzeuges UniMoDoc wird ein Dokumentationschema für diese Produktionslinien-Software ausgearbeitet und für einen ausgewählten Teil der Software in Form einer Nachdokumentation angewendet. Mit der erstellten Dokumentation wird eine Fallstudie durchgeführt, welche die Nützlichkeit des Dokumentationsschemas überprüft. Zusätzliche Interviews zur Fallstudie ermöglichen eine Evaluation des Dokumentationsschemas, die zum Ergebnis der Bachelorarbeit führen. Diese Bachelorarbeit zielt auf eine abstraktere Form der Dokumentation ab, um die übergreifenden Funktionsweisen von Software Modulen zu verstehen. Es wird kein reines Quellcode Dokumentationsverfahren behandelt.

1 Introduction

Software is used almost everywhere we can use it today. Money transfers, cars, ERP systems and telecommunication are one of many applications which are mainly controlled by software. These are in fact very critical applications where a high standard of software quality is essential. Today software products for most applications are very complex and splitted into multiple parts. Developing such software products takes place in different kind of software developing processes. In every software developing process we have analysis, design, construction, testing and maintenance phases. The analysis phase identifies the requirements for a task. For example we want a car to brake automatically if the front car suddenly brakes and our car will not be able to brake without crashing after this moment. So we have different complex requirements the software must fulfill to handle such situation. These requirements will be identified in the analysis phase. The design phase decides how the task requirements will be implemented in the software. Questions like: "Which data will be processed? Which communications can be used? Do we already have such functionalities that we could use? Which design pattern could be useful?" will be solved in this phase. The construction phase implements the task. The software product is the result of this phase. In the testing phase the implemented task of the software product will be validated and verified. The validation ensures the software product works right that the car brakes automatically if the front car comes too fast too close and our car will never get in the situation that the distance is not enough to brake without crashing. The verification ensures that the software product meets the requirements which have been set in the analysis phase and the defined functionalities of the design phase are working as described. The maintenance phase takes care of occurring bugs or additional funcionalities after the software deployment. These phases can be repeated and partitioned differently in software development processes. So we see that there are lots of parts in a software development process where documentation will be necessary. The motivation of this thesis is an attempt to improve the documentation

during the development process. [IEEE, 1990]

1.1 Motivation

Developing complex software systems requires a high level of information. In a developing process the resulting information from each developed software module needs to be shared and stored for future use or related processes. For a complex software system consisting of multiple modules the function of each module and the communication between the modules have to be clear for every software developer working on existing and additional modules. These information are required to understand the whole software system and to allow an efficient and high qualitiy development. A high level developer documentation is required to achieve this state of knowledge.

A high qualitative developer documentation contains structured and well-defined information about the related software module. An easy and smart procedure for creating software module documentation motivates involed software developers and supports software developers maintaining the documentation throughout the process.

Previous studies at the Institute of Software Technology (ISTE) considered the significance of software module documentation especially during the whole software development process. They identified different levels of software module documentation which will be discussed at chap.3 - State of the Art. The results mainly focus on the benefit of the software module documentation. This is the most important fact for the research of software module documentation because the effort for documentation needs to be justified and the approval for this effort needs to be forced by the engineers. So we keep in mind that we have to consider more psychological aspects in this thesis than it has been done in the previous studies. [Kuhn, 2012, Kircher, 2012, Strobel, 2012]

The industrial partner of this thesis is a software production company with a major business devision for Production Line Systems. They remark big deficits in their software module documentation and they are currently in an ongoing improvement stage of their software development process. Their software system is complex, wide-ranging and mostly customer oriented. In their opinion it is quiet difficult to generate detailed documentation but we will discuss this more precisely in the

analysis part. The current issues, the industrial partner is facing in their software development process, are a great opportunity for the evaluation and motivation for this thesis.

Other theses at the ISTE relating to previous theses have already prepared a module documentation scheme. This module documentation scheme was embedded in their documentation software UniMoDoc which was created in the course of these theses. This thesis uses the results of these theses and extend them. [Pankratz, 2013, Casciato, 2013]

1.2 Goal

We think that the balance between abstraction level and benefit will manage a practicable method to create software module documentation. We need to introduce a documentation method which easily provides a compact view on the software module functionalities and does not occupy too much time during the software development process. Software developers should not notice big differences with or without parallel documentation. Documentation has to be treated as indispensable in software development processes of complex software systems. The idea of this thesis in reaching that treatment, is creating a clear and compact documentation structure to allow easy and direct access to the information on the one hand and on the other hand to assure fast and comfortable possibilities storing information.

In this particular case we develope a module documentation structure that can be used by the industrial partner. The module documentation structure can be used in a productive environment and will improve and support the software development process where it is used. Therefore we also want to identify the reasons and the needs of the industrial partner. Afterwords we should be able to reveal the benefit of this module documentation structure in the productive environment.

1.3 Strategy

In the beginning of the thesis, the research of the ISTE and the situation at the industrial partner was analysed. The task was to investigate the state of the art for module documentation that also includes the previous theses of the ISTE to use that knowledge. One software product of the industrial partner was selected by the industrial partner for this thesis. This software product contains lots of software modules but only specific software modules were chosen because there were to many to consider all of them in the time period of this thesis. The selection process of the software modules is described in chap.4 - Documentation Analysis. The documentations of the chosen software modules were analysed and evaluated in this thesis. The evaluation results in the documentation type, documentation quality and the abstract parts which are used in the documentations. The identified scheme of the module documentation of the industrial partner was merged with the scheme of UniMoDoc as the next step of the thesis. The final case study evaluates the authentic use of the developed documentation scheme. Therefore an existing documentation was transformed into the developed documentation scheme in form of a UniMoDoc document. The results of this case study evaluate the developed documentation scheme.

1.4 Outline

This thesis is structured as follows:

Chapter 1 - Introduction: Introduces the task and ambition of this thesis.

Chapter 2 - Terms: Important terms that are used in this document will be described here.

Chapter 3 - State of the Art: Existing results and artifacts that relate to this thesis will be introduced here.

Chapter 4 - Documentation Analysis: The information about the industrial partner and the results of the analysis are presented here.

Chapter 5 - Scheme Matching: The realization of the module documentation scheme matching and the scheme itself are introduced here.

Chapter 6 - Case Study: The executed case study of this thesis and its intention are introduced here.

2 Terms

In this chapter the important terms will be described. They will be used all over this study.

2.1 Software

A part or a whole program which is defined by different procedures and rules. It is integrated in an information processing system. Documentations which describe functionality, behaviour or usage are also associated.

[ISO/IEC/IEEE, 2010, § 3.2741]

2.2 Software Quality

"Capability of a software product to satisfy stated and implied needs when used under specified conditions."

[ISO/IEC/IEEE, 2010, § 3.2780]

2.3 Software Module

An independent software which is a part of a standalone software. It is designed to be integrated in different software so it is reusable. The integration is described with an interface.

[IEEE, 1990, p. 49]

2.4 Software Development Process

Requirements of an user are used for analysis, design and implementation to create a software. The creation phase also includes a testing phase that verifies that the software meets the user requirements. Another phase of the process is the installation of the software at the user's environment. There are different kind of processes how to handle each phase of the process. Some of these processes are incremental development, rapid prototyping, spiral model or waterfall model.

[IEEE, 1990, p. 67]

2.5 Software Maintenance

Activities modifying already delivered software are to correct existing faults, improve performance or change other properties which do not extend the software with new functionalities. Software maintenance can be splitted into

- adaptive maintenance: Modifies the software to be compatible with a different environment
- corrective maintenance: Fault correction in the delivered software
- perfective maintenance: Performance or maintainability improvements of the delivered software
- preventive maintenance: Prevention methods in the software development process before the software will be delivered to prevent activities of adaptive, corrective or perfective maintenance

[IEEE, 1990, p. 8, 22, 46, 55, 57]

2.6 Software Documentation Tool

The software documentation tool supports the software engineer to store the information about functionalities, properties and implementation of the developed software. Documenting the software in a specific software documentation tool should be a standard procedure for a software engineer.

[IEEE, 1990, p. 67]

2.7 Developer Documentation

A documentation which is mostly created by software developers. It is only created to help software developers to understand the abstract behavior of the described software.

[IEEE, 1990, p. 28]

2.8 UniMoDoc

The Universal Module Documenter is a software documentation tool which was developed for the ISTE during the theses [Pankratz, 2013, Casciato, 2013].

3 State of the Art

3.1 UniMoDoc

The Universal Module Documenter is a software documentation tool.

The UniMoDoc application which is used in this thesis is a Java Swing application that runs on the JVM. It allows the developer to create and use his own documentation scheme. It supports reference visualization and a few data fields. It is a good choice for an independent software documentation with highly customizable requests.

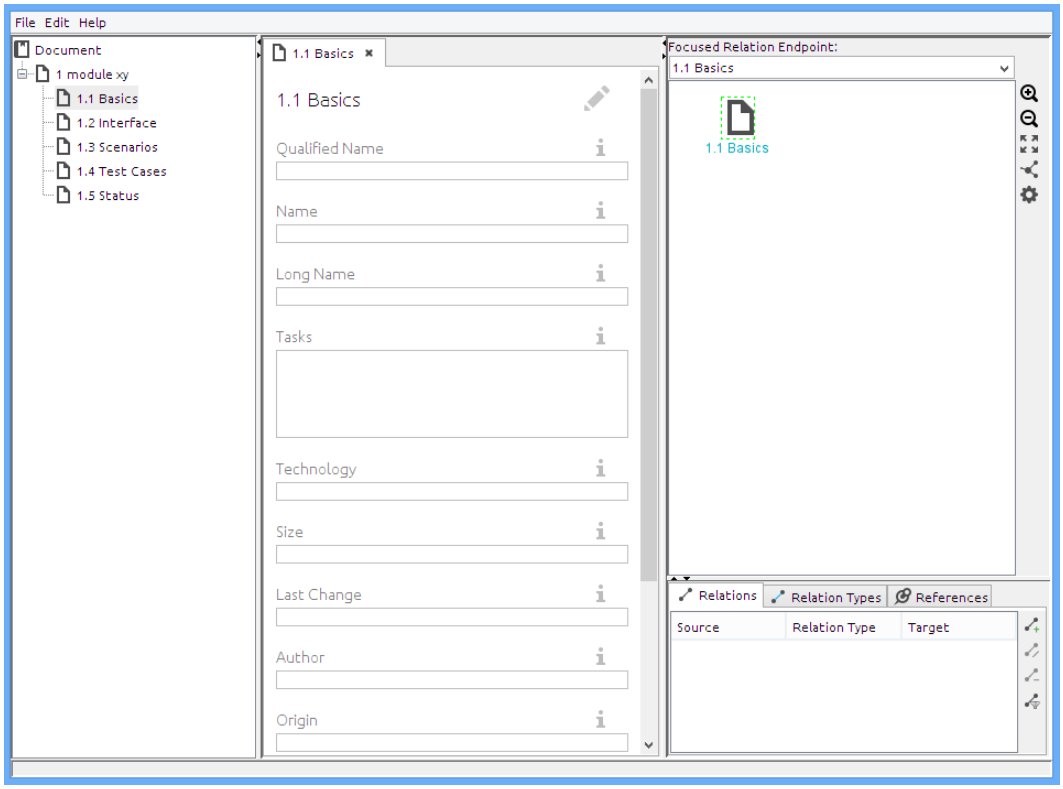

Figure 3.1: UniMoDoc 1.01 (Swing with standard template)

The project was initiated by Ivan Bogicevic, but the Swing application project was only private and will not be continued. The UniMoDoc application which is not used in this thesis, is a web application. The tool was published as a Sourceforge project by Ivan Bogicevic. [Bogicevic, 2014]

4 Documentation Analysis

As written before the industrial partner was purposely selected because besides the analysed software modules, the industrial partner is running further continous developed software systems. The first step of the thesis was to communicate the current software state with the industrial partner and to discuss which software system will be considered in the thesis. Finally we chose the software system "Line Browser Module Server" which is separated in several independent software modules. The LBMS processes data of a production line through a lot of different modules. Each module has a specific funtionality. There are modules responsible for processing data of third party hardware tools like wrenches, scanners, cameras or PLCs. There are modules responsible for processing data of other modules like ensuring data persistance on databases or file systems. All of these modules and their different functionalities make this system very complex. The communication between the modules is based on messages. The modules and therefore the functionalities of the modules can be controlled or activated by messages.

4.1 Selection

The Production Line Software of the industrial partner contains more than 100 interacting modules. A lot of these modules connect external devices to the production line software. 22 of the modules are considered in this thesis. The modules were randomly chosen by the following criteria for their documentations:

- Complexity being defined by a companies software architect
- Existing documentation of the software module
- Importance in the Production Line Software
- Satisfaction of the existing documentation scheme

- Comprehensible description of the module behavior inside the Production Line Software

The following modules were chosen for this analysis:

DPerformer, EA Server, Envelope Collector, File Observer, FisPdaStatus, FmstPerf, Ident, IdentCreator, ManualScrewdriver, PickToLightController, PrintOut, ProcessflowPerformer, ProductionController, Rework, Rework Statistic, Screwdriver, mod_printer, halconCamera, SequenceInfo, SpsPcv3, WorkerIdent, LBMS (Kernel)

4.2 Status

The following modules were chosen for this thesis. The relating status for every module represents the detail level of the existing documentation and can be low, medium or high in ascending order of detail level. Each status includes the properties of the status before. For example a module with status medium also achieved the properties of the status low. Additional documents or information like UML diagrams, external specifications or protocols improve the status.

The status low could be achieved if the module documentation gives a comprehensible description about the module functionalities.

The status medium could be achieved if the module documentation contains informations about module configuration and communication.

The status high could be achieved for described module behavior scenarios which can be described in free text or any form of visualization like UML diagrams. This status could be also achieved if external specifications or protocol descriptions were available for the module.

The following table shows the status result of the selected module documentations:

Module	Documentation status
DPerformer	low
EA Server	medium
Envelope Collector	high
File Observer	low
FisPdaStatus	medium
FmstPerf	high
Ident	high
IdentCreator	low
ManualScrewdriver	low
PickToLightController	medium
PrintOut	low
ProcessflowPerformer	high
ProductionController	low
Rework	low
Rework Statistic	low
Screwdriver	high
mod_printer	medium
halconCamera	medium
SequenceInfo	low
SpsPcv3	high
WorkerIdent	low
LBMS (Kernel)	high

Table 4.1: Software module limitation

4.3 Evaluation

The main module documentations are documented in Microsoft Word files. The documentation files are stored in an Apache Subversion which is a version control system [CollabNet, 2014]. The developers could use an existing Microsoft Word template in English and German with a predefined documentation scheme for the creation. But this was only possible for module documentations that had been created after the creation of these MS Word templates. Previously created module documentations had no templates available, so they could not underlay any documentation scheme. Additionaly all the documentations are differently structured because they were written in free text. The unstructured software documentations impede an efficient training of new software engineers or existing software engineers which are not familiar with the related software. Additional information are stored in image files, XML files and PDF files. Most of the image files are UML diagrams that were exported as image files. The diagrams describe functional scenarios of the modules. Example configurations and messages are stored in XML files or pasted in the MS Word document. External specifications and protocol descriptions are stored as PDF files.

The communication through messages respectively the configured messages and their functional effects are described in most of the module documentations. The module communication and configuration are the most documented information. The level of information details can vary strongly between the different module documentation.

4.4 Identified Scheme

Through the analysis of the selected modules the following scheme was analysed in this thesis. The following listing shows the identified documentation scheme of the selected module documentations:

1. Introduction
2. Configuration
    a) Fully qualified name element
        i. Attribute
        ii. Description
        iii. Type
        iv. Default Value
3. Messages
    a) Inbound
        i. Messagename
        ii. Description
    b) Outbound
        i. Messagename
        ii. Description
4. Terms
    a) Term
    b) Description
5. Protocols
6. Embedded external specifications (e.g. connected devices, signals)
7. Scenario description
    a) Sequence diagrams

5 Scheme Matching

With the identified documentation scheme of the selected module documentations, the UniMoDoc documentation scheme can be matched with it. The non existing elements in the UniMoDoc documentation scheme which exist in the identified documentation scheme are added to the matched scheme. Most elements from the UniMoDoc documentation scheme are added to the matched scheme. The "Scenarios" and "Test Cases" elements are modified in the matched scheme. This modification was built in consultation with the industrial partner. Also non existing but required elements in each existing scheme are added to the matched scheme.

5.1 Existing UniMoDoc Scheme

The thesis refers to the UniMoDoc Version 1.01. The refered UniMoDoc Version is deployed with the MODULE-SPECIFICATION.XML including the following module documentation scheme. The UniMoDoc documentation scheme is splitted into 5 global parts.

1. **Basics** Basic information about the module are described.

2. **Interface** Available public communications with the module are listed.

3. **Scenarios** Module behaviors for specified communications are described.

4. **Test Cases** Correct and failure behaviors of the module are described.

5. **Status** Tracking option for the documentation status.

The following listing shows the UniMoDoc documentation scheme in detail:

1. Basics
    a) Qualified Name
    b) Name
    c) Long Name
    d) Tasks
    e) Technology
    f) Size
    g) Last Change
    h) Author
    i) Origin
    j) Tools
    k) Source
    l) Starting Point

2. Interface
    a) Public Interface
3. Scenarios
    a) Standard Scenarios
    b) Failure Scenarios
4. Test Cases
    a) Standard Test Cases
    b) Failure Test Cases
5. Status
    a) Status of Implementation
    b) Review Status of Implementation
    c) Status of Documentation
    d) Review Status of Documentation

5.2 Scheme Matches

The identified documentation scheme of the selected module was matched with the UniMoDoc documentation scheme to this resulting documentation scheme. The UniMoDoc documentation scheme was taken as a basis and complemented with the industrial partner's documentation scheme. Some parts of the matched documentation scheme were extended with important properties on agreement with the industrial partner.

The documentation scheme of this thesis is splitted into 8 global parts.

1. Basics Basic information about the module are described.

2. Configuration Settings for the module configuration are set in elements with parameters and attributes in this documentation scheme.

2. b) Work This is a special feature for the modules of the industrial partner. A self made script language can send messages with variables. The messages can be set with variables of incoming messages in the configuration of each module. The work element in the documentation scheme contains the variables which can be set in each configuration.

3. Interface Available public communications with the module are listed.

4. Messages Inbound and outbound messages that can be received or sent by the module are documented in this part. It allows the developer to describe the functionality of the message and its parameters.

5. Scenarios Module behaviors for specified communications are described. This can be failure or standard scenarios. It is possible to attach UML diagrams or other visualizations of the usage.

6. Test Cases The existing and missing test cases give a quick overview over the module test status.

7. Status Tracking option for the documentation status.

8. Protocols Used protocols in the module implementation are documented.

9. External specifications For example specifications for acquirable data from third party hardware tools like printers, wrenches, handheld barcode scanners or conveyor controls.

The following listing over the next 4 pages shows the resulting documentation scheme of this thesis in detail:

1. Basics
    a) Qualified Name
    b) Name
    c) Long Name
    d) Tasks
    e) Technology
    f) Size
    g) Last Change
    h) Author
    i) Origin
    j) Tools
    k) Source
    l) Starting Point

2. Configuration
    a) Element
        i. Fully qualified name
        ii. Format
        iii. Attributes
            A. Name
            B. Type
            C. Description
            D. Default Value
            E. Possible Values
        iv. Parameters
            A. Name
            B. Type
            C. Description
            D. Default Value
            E. Possible Values
    b) Work
        i. Command
        ii. Comment
        iii. Accessable variables
            A. Name
            B. Type
            C. Default Value
            D. Possible Values

3. Interface
    a) Public Interface
4. Messages
    a) Inbound
        i. Messagename
        ii. Description
        iii. Comment
        iv. Parameters
            A. Name
            B. Description
            C. Type
            D. Default Value
            E. Possible Values
        v. Senders
            A. Modulename
    b) Outbound
        i. Messagename
        ii. Description
        iii. Comment
        iv. Parameters
            A. Name
            B. Description
            C. Type
            D. Default Value
            E. Possible Values
        v. Receivers
            A. Modulename

5. Scenarios
    a) Standard Scenarios
        i. Introduction
        ii. Procedure
        iii. State
        iv. Diagram
    b) Failure Scenarios
        i. Introduction
        ii. Procedure
        iii. State
        iv. Diagram
6. Test Cases
    a) Existing Test Cases
    b) Missing Test Cases
7. Status
    a) Status of Implementation
    b) Review Status of Implementation
    c) Status of Documentation
    d) Review Status of Documentation
8. Terms
    a) Term
    b) Description
9. Protocols
10. External specifications

The matched documentation scheme is abstract. The way to represent this documentation scheme in a document or a documentation tool is given in this thesis. But the following elements can occur mutiple times in any kind of representation:

- Accessable variables
- Attributes
- Element
- Parameters
- Standard Scenarios
- Failure Scenarios
- Inbound
- Outbound
- Terms
- Senders
- Receivers

The following figure shows a sample mapping and multiplicities of the documentation scheme in UniMoDoc:

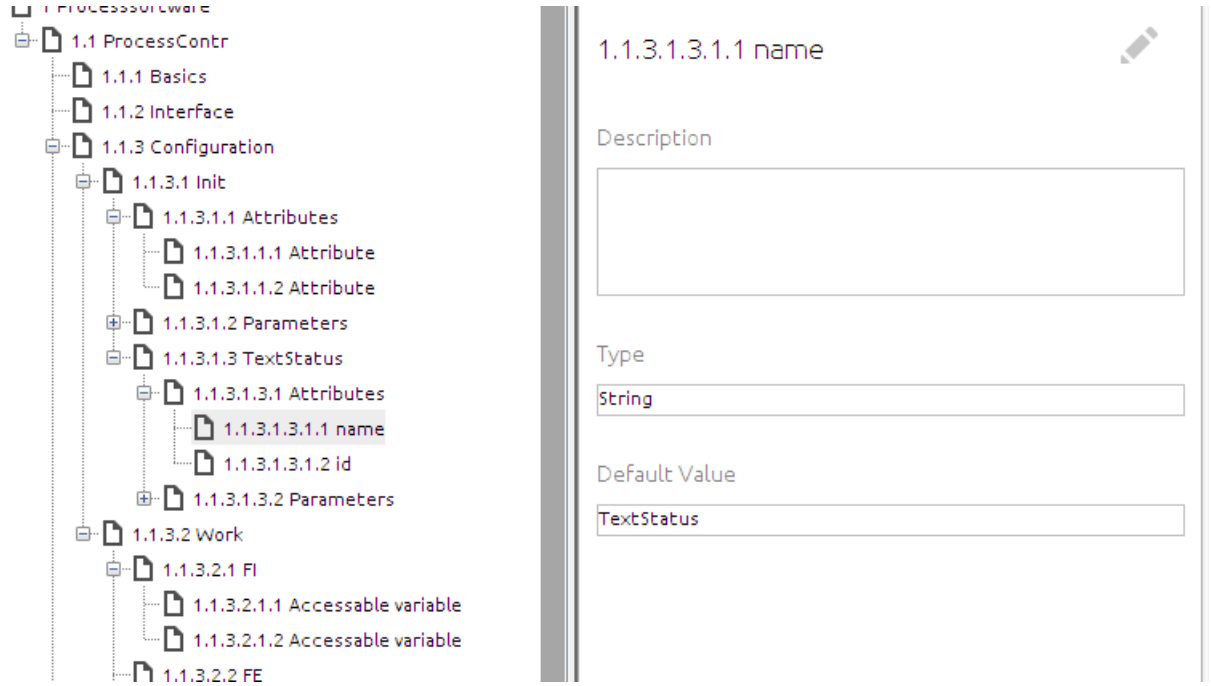

Figure 5.1: Example representation - Configuration - Element (Init) - Work

5.3 Identified UniMoDoc Issues

The matched scheme was implemented in different UniMoDoc templates. The created UniMoDoc templates are required for chap.6 - Case Study of this thesis. It is used to transfer an existing documentation into an UniMoDoc document using the matched documentation scheme.

During template creation and the research about the UniMoDoc documentation scheme, the following issues in UniMoDoc were identified:

- Missing widgets (fields)
  - DropDown
  - Tables
  - UML diagrams
- Missing functionalities
  - Drag'n Drop of files directly in UniMoDoc (UniMoDoc projects, images, text documents)
  - Templates should be integrated somehow in the project file. They are currently stored locally and need to be copied manually to new environments
- Usability changes
  - Enter button in edit fields should accept changes and switch to next fields
  - Infofield button which describes the functionality of a field should not block the process. It can only be closed by mouse click. Could be more a mouse hover effect.
  - Template management should replace new saved templates with same name. Otherwise there are more templates with the same name and not identifiable.

6 Case Study

The thesis evaluates the matched documentation scheme in this chapter. It is important getting feedback from the industrial partner for the documentation scheme. As in the book of [Runeson et al., 2012] described

> "Case studies investigate phenomena in their real-world settings, ..."

the thesis can investigate its documentation scheme in this case study for real-world settings. An existing module documentation was transfered into the documentation scheme of the thesis into UniMoDoc. The industrial partner used this transfered module documentation and was interviewed at the end to evaluate the utility and usability.

6.1 Preparation

In the beginning of the case study, one of the LBMS modules was chosen to be used in the case study. The industrial partner selected one module with a real problem. A derivate of this module was developed before but the industrial partner wanted to merge the main module and the derivate. The module documentations from "AudiFmstPerf" and "FmstPerf2" modules needed to be merged to create a new module or implement the missing functionalities of one into the other. Because of the outdated documentation, they were updated at first from the industrial partner. The industrial partner needed to work with the merged and transfered documentation in UniMoDoc. To evaluate the case study, a survey was prepared for an interview with the involved employees of the industrial partner.

6.1.1 Participant Survey

The case study participants were asked questions before and after they used the new documentation scheme. So we received answers based on the old documentation scheme from the industrial partner and we received answers based on the documentation scheme of this thesis. As a result we evaluated the usability and utility of the thesis documentation scheme.

The questions were the same for every interview. The main questions aimed the basic knowledge of the software engineer and the special questions aimed the opinion of the documentation schemes.

- Main
  - How long do you work as a software engineer?
  - Which technologies have you applied before?
  - Over which time period have you applied the technologies?
  - Which software development processes have you applied before?
  - Which documentation methods do you know?
  - Which of these methods do you use commonly?
  - How do you proceed at documenting a software module normally?
  - When do you make changes in a module documentation?
  - What advantages do you see for keeping a documentation up to date?
  - What challenges do you see in software documentation in general?
  - Which elements are important to you in software documentation?

- Special
  - Assume you would be assigned for a development task for a module you do not know. How do you proceed?
    - Why do you proceed as described?
  - Which properties respectively components are offered in the documentation scheme which support you creating a module documentation? Why?
  - Which properties respectively components are missing in the documentation scheme to create a module documentation? Why?
  - Which properties respectively components of the documentation scheme are conducive to the comprehension in your opinion? Why?
    - In what way does the documentation scheme ease acquiring information of the module?
  - Which properties respectively components are missing in the documentation scheme which would be conducive to the comprehension? Why?
  - Which properties respectively components are obsolete in the documentation scheme ?
  - Do you prefer a module documentation in free text? Why?

6.2 Procedure

The following steps reveal the procedure of the case study.

1. Preparation
    a) Select eligible software module/modules
    b) Two employees of the industrial partner rework the outdated documentations
    c) Create case study participant survey
    d) Merge the module documentations in the thesis documentation scheme into UniMoDoc
2. Interview employees which worked with old module documentations and know the existing documentation scheme of the industrial partner
3. The employee works with the module documentation in UniMoDoc
4. Interview the employee who worked with the module documentation in UniMoDoc
5. Evaluate the interview results

6.3 Evaluation

6.3.1 Interviews

The interviewed employees have similar skills. They are all familiar with technologies C++, Java or Delphi. Known software development processes are scrum, extreme programming and test driven development. Documentations are created as MS Word files, UML diagrams, code comments or hand written notes. Code comments are made before a separate documentation is created. Documentations are changed for functional changes in the software module. The employees have the consistent opinion that keeping the module documentation up to date saves time because software engineers do not have to debug the module or read code comments multiple times. It also prevents other software engineers from making mistakes. The challenges for module documentations are to keep it consistent, comprehensible and up to date. Another challenge is the interpretation of free text writing.

Important elements of the interviewed employees for a module documentation are:

- Sequence diagrams
- Class diagrams
- Procedure descriptions
- Important procedures
- Interfaces
- Configuration
- Overview
- Input / Output messages

The existing documentation scheme is vague according to the answers of the employees. The information need to be more abstract to understand the global functionalities. A unified representation of information increase the comprehensibility. It is also important to understand the communication between the modules so the module connections should be better described. Scenarios of the communication and the module functionalities improve the efficient understanding. Diagrams could visualize the scenarios for an easier point of view. More information about protocols and external specifications are required in the module

documentation. The documentation scheme of this thesis could be extended with UML diagrams, more information about architecture and libraries.

6.3.2 Work with the Documentation Scheme

One employee worked with the documentation scheme in UniMoDoc. He got a UniMoDoc document which included the documentation scheme of this thesis. The documentations of the two selected software modules were already merged into that UniMoDoc document by the thesis processor. The task was to get in touch with the documentation scheme, the documentation tool UniMoDoc and to complete the module documentations. The task to complete the documentation was decided with the industrial partner.

The employee understood the documentation scheme without any problems. After a short time period he was familiar with the existing UniMoDoc document and the available templates in UniMoDoc which allow the usage of the documentation scheme. In cooperation with the module source code the employee added details to the existing configuration and messages. He also added new configurations, messages and scenarios. He reported a positive attitude towards UniMoDoc and the documentation scheme. The employee noted usability features for UniMoDoc which had nothing to do with the documentation scheme. In summary, he demanded a more supporting role for the documentation tool. The following figures are part of the resulting UniMoDoc document.

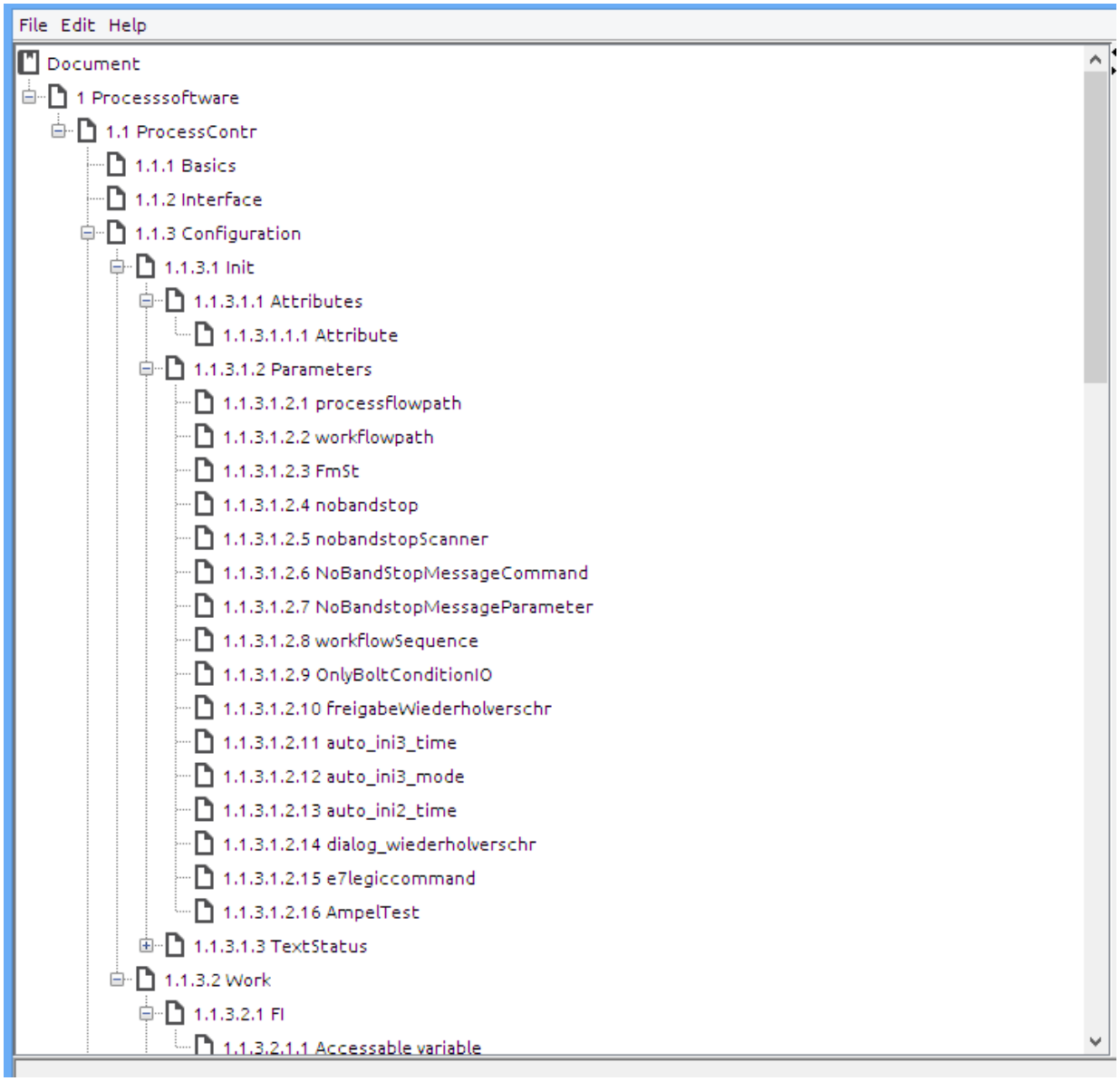

Figure 6.1: Configuration part of the case study document

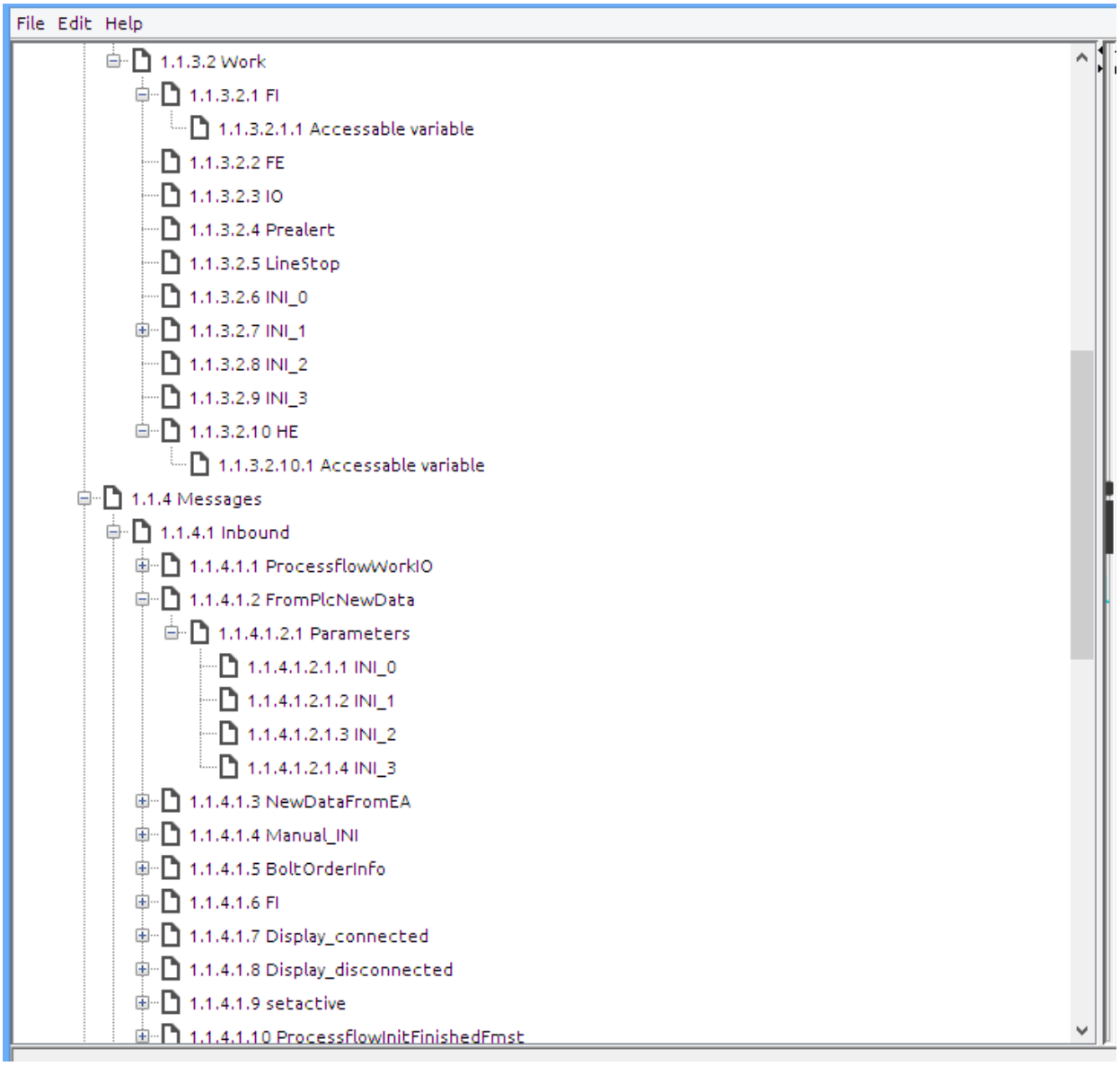

Figure 6.2: Message part of the case study document

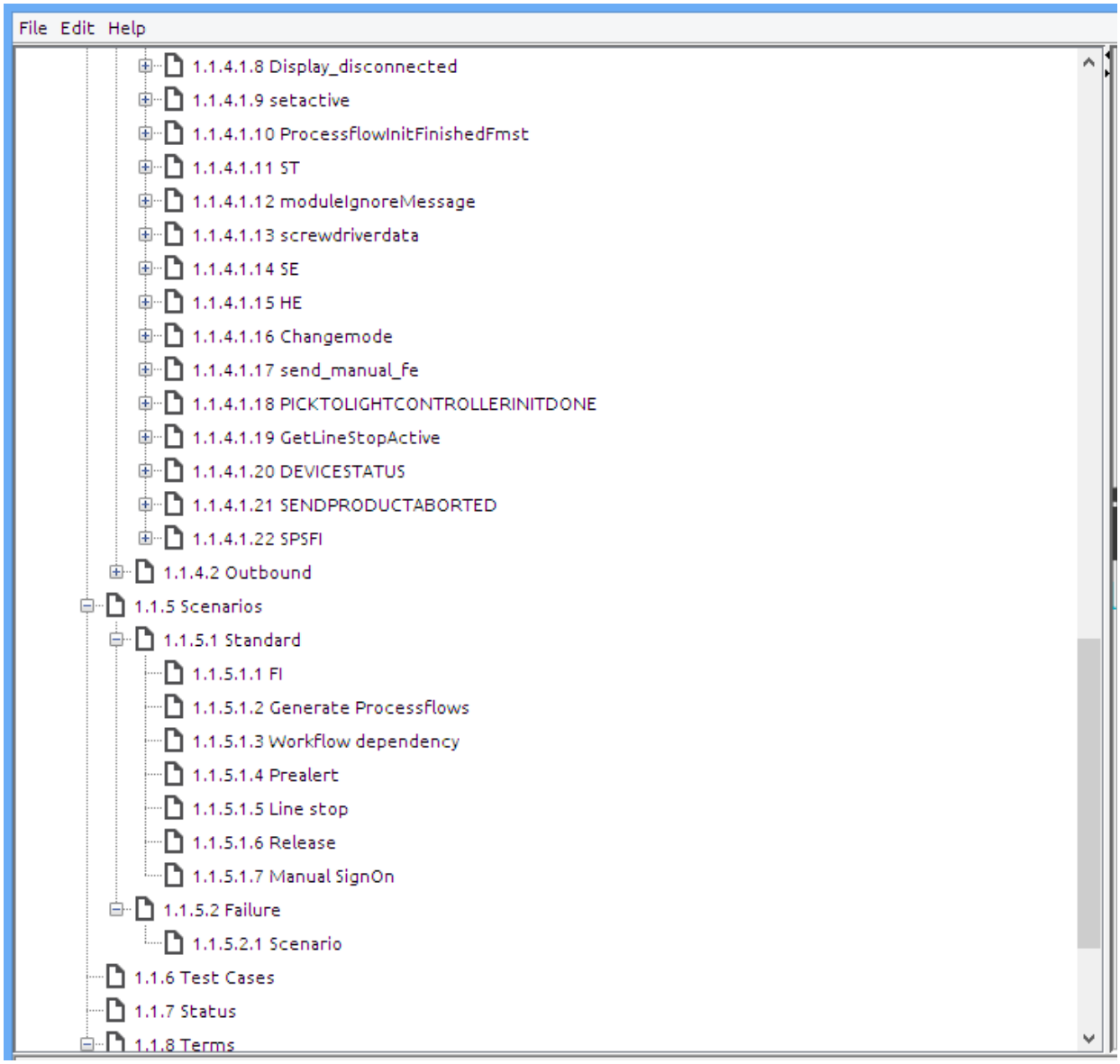

Figure 6.3: Scenario part of the case study document

6.4 Conclusion

Based on the interviews with the employees and the practical application of the documentation scheme, the thesis goes in the right direction. In this case, the industrial partner is searching for a solution that the thesis can provide. A structured documentation scheme is demanded but in combination with a restricting environment. A documentation scheme in combination with a free environment like MS Word is not practical considering the straightforward content. The utility of the documentation scheme is only available in a restricted environment. The usability of the documentation scheme is defined by the consistency and the support of the assigned environment. Today Software development tools offer a great automatic support in many ways of directions. This is also required for a supporting software documentation tool which can represent a usable documentation scheme.

7 Prospect

The role of module documentation needs more attention in the near future. Documentations need the same requirements to structure and restriction other parts or processes of software development already have. The documentation scheme in this thesis is an example for a structured approach but it can be extended or reduced for a specific domain. Software documentation tools like UniMoDoc can help you achieve a customized documentation process in combination with a documentation scheme but it also needs to be more usable. More complex systems in software engineering demand more abstract and custom documentations which brings us to a new generation of software documentation tools.

Bibliography

Ivan Bogicevic. Unimodoc sourceforge project, June 2014. URL `http://sourceforge.net/projects/unimodoc/`. (Cited on page 22.)

Davide Casciato. *Verwaltung von Testinformationen in der Moduldokumentation*, page 95. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2013. (Cited on pages 13 and 19.)

Inc. CollabNet. Apache Subversion, June 2014. URL `http://subversion.apache.org/`. (Cited on page 26.)

IEEE. IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard, September 1990. (Cited on pages 12, 17, 18, and 19.)

ISO/IEC/IEEE. Systems and software engineering – vocabulary, December 2010. First edition. (Cited on page 17.)

Michael Kircher. *Integrierte Dokumentation für Software-Module*. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2012. (Cited on page 12.)

Tobias Kuhn. *Verbesserung eines Dokumentationswerkzeugs für Java-Pakete*, page 72. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2012. (Cited on page 12.)

Dimitrij Pankratz. *Tool Support for Software Architecture Documentation*, page 93. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2013. (Cited on pages 13 and 19.)

Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering - Guidelines and Examples*. Wiley, 2012. ISBN 978-1-118-10435-4. (Cited on page 39.)

Patrick Strobel. *Erfassung von Anforderungen an Software-Module.*, page 62. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, 2012. (Cited on page 12.)

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.

Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.

Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.

Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort / Datum / Unterschrift

Declaration

I hereby declare that the work presented in this thesis is entirely my own.

I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.

Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.

The electronic copy is consistent with all submitted copies.

Place / Date / Signature