IAS-Regulations Manual v6.0/English

Prof. Dr.-Ing. Dr. h.c. Michael Weyrich
Universität Stuttgart
Institute of Industrial Automation and Software Engineering
Pfaffenwaldring 47
70569 Stuttgart

http://www.ias.uni-stuttgart.de

Foreword

The Institute of Industrial Automation and Software Engineering (IAS) uses a quality management system which follows the guidelines given by DIN EN ISO 9001. Of all available norms for certification (9001, 9002, 9003), this one is the most comprehensive, and it is applicable to all enterprise types.

The IAS considers itself as a service enterprise for its customers; these are the students, staff members, industry partners and last but not least the state, or rather the society, which it is committed to as part of the educational institution called university.

The IAS offers the following services:

- Education of students in teaching and research in lectures, courses, internships and by advising students during thesis work;
- Providing a professional work environment;
- Advisory during dissertation work;
- Cooperation with industry enterprises on research work;
- Scientific exchange with other universities in Germany and foreign countries.

In exchange, students and staff members perform services in the form of thesis work, dissertations and other scientific activities for the IAS.

In the framework of a process-oriented quality management system at the IAS, all important work processes and services are recorded; they are represented in the form of procedure rules in the Quality Management Manual (QMM) and they are supervised by applying suitable instruments, e.g., surveying people involved in the work process. Check measures are used to register deviations from target goals, to recognize improvement possibilities in the representation and execution of work processes and to implement corrections accordingly.

During thesis projects at the IAS the students learn to proceed according to a pre-defined process, to secure the quality of their work and to evaluate it together with their supervisors. The process of thesis work is defined in the IAS process model, which offers 4 process model classes, and is thoroughly illustrated in this IAS Regulations Manual. Then a description of the IAS Development Environment (DE) will follow in which all necessary tools for the execution of thesis projects are explained. Supplementary to the IAS Regulations Manual the procedures for thesis works are also described in the form of procedure rules in the QMM. The process model classes are also summarized in the QMM and integrated in the AM procedures (Bachelor Theses, Research Theses and Master Theses).

General Information

IAS Homepage "*www.ias.uni-stuttgart.de*"

The IAS homepage provides current information about events at the IAS, teaching, research and staff members. In particular, it contains information on lectures, exercises and the colloquium, where students present the results of their diploma or master theses. Additionally, research topics of the IAS are listed and described.

IAS Network Directory *ias-cifs.tik.uni-stuttgart.de**studdir*

This directory includes all document templates required for thesis' projects. The following sub-directories are important:

.\StAr-Vorlagen & .\StAr-Templates	document templates for the IAS process model (German and English)
.\Richtlinien – Regulations Manual	IAS guidelines for work on student projects, description of the IAS development environment and programming guidelines

Hint:
Templates and documents are updated from time to time. Please make sure that you always use the most recent version of templates or documents. Any updates or changes will be announced per e-mail.

Project Directory (drive S:)

Project directory for thesis projects (only the supervisors have writing access).

Lockers for IAS-Students

Students can apply for a key for one of the lockers in the laboratories. Contact your thesis supervisor.

Access to IAS Directories

At the start of the thesis, the student's supervisor will ask the system administrator to provide an account for the student. As soon as the account has been set up, the student receives an e-mail including a description on how to include the IAS directories to the student's account. For the first login, enter the username at an arbirtrary IAS computer while leaving the password empty. During the login process, you will be asked to provide a password.
After this first login, IAS students have the additional option of accessing the IAS directories from an external device. For this, establish a VPN connection to the University

of Stuttgart. Then, you can then access the directories via the file explorer. When asked for your password, provide your collegiate e-mail address and the corresponding password.

Table of Contents

1. IAS Boundary Conditions

1.1 Prerequisite to Begin a Thesis

Each student project must be registered with the Office of Curricula (Prüfungsamt). Bachelor and Master theses can only be commenced with after the respective required number of credit points have been achieved.

In the case of **Bachelor and Master theses**, the **registration of the thesis at the IAS** with the theses' coordinator (TC) must take place **before the initial meeting** with Prof. Weyrich. The **print-out from C@mpus for the registration with the Office of Curricula** must be taken along to the initial meeting with Prof. Weyrich.

At the beginning of thesis work, the student will receive a sign-off sheet (inter-office slip) together with these regulations. The sign-off sheet will be used to record and confirm the progress of the thesis. Upon finishing the thesis, the sign-off sheet will be submitted along with the thesis.

1.2 Project Duration

The effective project duration for Bachelor Thesis (equal to Bachelorarbeit) (BT) is 3 months, for Research Theses (equal to Forschungsarbeit) (RT) 3.5 months, for Master Theses (equal to Masterarbeit) (MT) 6 months, and for Study Theses 3 months. These numbers are set accordingly to the examination regulations of the respective study program.

At the beginning of the project the student and his supervisor will agree on a preliminary deadline (foreseeable interruptions, e.g., exam preparation, internships, etc. are to be taken into consideration). The closing date must be authorized by the director of the institute. Prolonging the set deadline is an exception to the rule and must be well-reasoned. In such cases, an extension of the deadline must be formless petitioned. The petition must contain the reasons for the extension, a new deadline and must be submitted to the supervisor. The supervisor will record the requested deadline on the sign-off sheet. The director of the institute will then decide on approval of the extension. A punctual execution will influence the grade of the thesis.

1.3 Laboratory Rules and Safety at Work

1.3.1 Laboratory Rules

When beginning the work at the institute, these regulations as well as the current laboratory rules, which are available from Mr. Villing, are to be read. All rules in theses documents must be adhered to. In addition, the following points must also be observed:

- Unauthorized persons are not allowed to enter the laboratories.
- Only such students are allowed to work in the laboratories that are completing a thesis, working as a scientific assistant or are participating in the laboratory course software engineering.
- In general, students are only allowed to work in the laboratories as long as IAS staff is present. An exception to the rule can be made by a staff member. A staff member must give a student written permission to remain in the laboratories although no IAS staff is present.
- No other persons are allowed access to the building or the laboratories unless they have the written permission of an IAS staff member.

- Anyone present in a laboratory after 5 p.m. must sign on the late shift sheet. This is also mandatory for persons who have begun their work in the laboratory before 5 p.m.
- When working in the laboratory on weekends, one must also sign on the late shift sheet before beginning work.
- Whoever leaves the laboratory last must turn off the lights and close all windows and doors.
- The laboratories may not be left unattended for longer periods of time. This is especially the case if work is continued on another floor of the building. In this case, the laboratory doors are to be closed.
- The fulfillment of these duties will be checked. If a person is found who does not fulfill such duties, then this person may be barred from working in the laboratories after 5 p.m.
- If irregularities are noticed, they are to be reported immediately to a staff member of the institute or to the director of the institute himself.

1.3.2 Safety at Work

The university is responsible for the safety of its employees and students at work. Each employee and student must get a safety introduction regarding potential hazards and measures to avoid these hazards. In particular, employees and students who work with electronic devices have to act with caution. Because of that, each student must attend the multimedia-based safety introduction and must acknowledge the successful completion on the sign-off sheet.

1.4 Procedures during Thesis Work

1.4.1 Project Begin

As a rule, the student will receive a detailed delineation (user requirements specification). Often the amount of time needed to complete work on certain subjects in research areas can only be determined at a later point in time. In such cases, the detailed delineation can only be completed during the project execution.

At the beginning of a thesis, a meeting with the director of the institute will take place in which the project tasks will be discussed and elaborated. During this conversation, the deadline for the project is defined. For this meeting, the students prepare a presentation to introduce their thesis' topic to the professor. A template for this presentation can be found in the templates' directory.

1.4.2 Procedures in Development Phases

The student projects will as a rule be completed as a project according to the IAS Process Model (IAS-PM). There is a process model class for each type of work (e.g., software or hardware) that will be defined in the delineation. A Research Thesis in the industry poses an exception to this rule, so a process model is not mandatory.

1.4.3 Computer Support

During thesis work, software tools are to be used as often as possible. The standard tools are described in the IAS development environment. Each supervisor is responsible for the training and the support of the student on certain tools. Changes in system software of PC's (e.g., PC configuration) are forbidden. After leaving the computer, the original settings (e.g., compiler options) are to be restored. When working on PCs, only files in the student's own directory (under z:\) may be changed (see also IAS directory structure).

Accounts for the computer network can be requested at your supervisor and will be assigned by the system administrator (room 1.138).

1.4.4 Report Duty upon Reaching Development Milestones

When development milestones have been reached, the supervisor must be informed of the intermediate results. The supervisor confirms these results by recording them on the sign-off sheet.

Problems or questions that arise during the project must be discussed with the supervisor as soon as possible. If errors or misunderstandings are discovered early, it is easier to make changes and helps to keep the project duration within the allocated time frame.

The strict adherence to this procedure influences the grade.

If a Research Thesis in the industry is not subject to the process model, a written report must be submitted to the supervisor every three weeks to depict the project's progress.

1.4.5 Intermediate Report for the director of the institute

After half of the time of a Study Thesis, Bachelor Thesis or Research Thesis, an intermediate report will take place with the director of the institute. This is also valid for Research Theses that are conducted without a process model. During a Master Thesis, two such reports will take place in 2-months intervals. The current status of thesis and documents are discussed, especially problems that have arisen and necessary deviations from the original schedule. As basis for the intermediate discussion, two or three slides which show the current state of the work (e.g. overview of the conception, software architecture, etc.) must be prepared. In the template folder, a PowerPoint template is available.

1.5 Supervisor

The supervisor(s) of a project (for undergraduate projects only one scientific staff person is necessary, for master theses the director of the institute and a scientific staff person will advise together) is (are) available for information and advice at arranged times. For each project, a further co-supervisor will be appointed to assist when the main supervisor is absent.

Reports on the progress are to be made at regular intervals to the supervisor of any undergraduate project or master thesis.

If for important reasons (e.g., sickness, exams, etc.) the project must be interrupted, then this must be reported to the supervisor immediately.

1.6 Presentation

The problem and the accomplished results of a Master Thesis project will be discussed in the form of a talk during the IAS colloquium. This presentation will be held four to six weeks prior to the deadline. The presentation is to serve as a final report and should contain the most important results of the project work. It also serves as a training for giving presentations. Hence, the students will be prepared and coached by their supervisor during test talks. A hardcopy of the slides is to be given to the director of the institute right before the presentation.

In contrast, the talk for Study Theses, Bachelor Theses, and Research Theses is to be held during the meeting of a students' group.

For preparing the presentation it is important to note the following points. Additionally, in the QMM, there is a document giving recommendations for a successful presentation in the IAS colloquium.

1.6.1 Structure of the Presentation

The presentation should contain the following points:

1 Motivation and overview (what will be discussed in the presentation?).

2 Presentation of the procedure and accomplished solutions (please describe the basic procedure only, a suitable selection of the solutions should also be made).

3 Problems during project work, a critical consideration of the results (were there special problems during the execution? How reliable are the results? How well was the posed problem solved, which issues might still be open? Here statements can be made as to which continuing projects may develop from of the thesis).

4 Summary (short final summary of the contents and the results in few sentences).

1.6.2 Presentation Method

1 The presentation should be held as candid as possible. The student should not just read from the manuscript. A manuscript only serves as an aid and to support the presentation.

2 A precise pronunciation is expected.

3 It is highly recommended to prepare slides with animations (presentation with video projector).

1.6.3 Presentation Duration

The duration of the presentation should be 20 minutes. It is strongly recommended to plan the presentation in such a way that the time limit is not exceeded.

1.7 Poster

The content of the thesis or the proect must be presented on a poster (size: DIN A3). This poster is used to present the thesis when handing it in. There is a PowerPoint template for the poster in the template directory.

1.8 Thesis Paper

Usually, text is to be processed in MS Word, graphics in Visio and the slides in PowerPoint. The use of any other formats must be permitted by the supervisor.

The thesis paper is to be done in such a way that it can be used as a draft for possible publication in a professional magazine. It should contain the following parts (see also the respective template in the template directory):

- Abstract (ca. a one-half DIN A4 page).
  The abstract should be done in both English and German (Zusammenfassung).
- A list of the abbreviations used.

- The length of the research work presentation and the results of project execution should be for:
    - Master Thesis: about 60-80 DIN A4 pages
    - Study/ Bachelor/ Research Thesis: about 40-60 DIN A4 pages

  The chapters and sections are to be structured in decades.

  Pages and pictures are to be numbered. All pictures are to include a caption. All pictures should be mentioned in the text e.g., ...(see Ill. 7)...“ or „...as shown in Ill.7“.

- References:

  The constraints on the layout of the template "literatur.doc", which can be found in the template directory, must be met. Additionally, there is a definition of the citation style in the guidelines.

1.9 Thesis Submission

The master thesis must be handed in to the director of the institute. Apart from your thesis paper, the following documents are necessary:

1.9.1 Poster

The results of the thesis must be explained using the poster.

1.9.2 Sign-off Sheet

Sections 1 and 2 should be filled out completely. During the final submission meeting, the submission will be confirmed.

1.9.3 Thesis Paper

Because the original thesis remains at the institute, the student should create a copy for himself.

1.9.4 Plagiarism Test

All student projects and master theses undergo a plagiarism test using the available software "PlagScan" at the institute.

1.9.5 Questionnaire

At the end of a master thesis or project, each student must fill out an anonymous questionnaire concerning the work at the IAS. The submission of the questionnaire must be confirmed on the sign-off sheet by the student.

1.9.6 Quality Management Certificate

Upon completing the study project, each student will receive a quality management certificate signed by the director of the institute and by the head of the IAS QM team. The certificate approves that the work was carried out according to the guidelines as defined in the IAS Process Model.

1.9.7 Follow-up

All unused directories and data on the PC's or SUNs are to be erased. The workplace should be emptied (private books/folders), books and keys belonging to the institute are to be returned.

1.10 Participating in the colloquium

Regular participation in the colloquium is part of the training of every IAS student and is expected during thesis and project work.

The colloquium takes place approx. every two to four weeks in the seminar room of the institute. Schedules are announced on the message board, on the IAS website and by e-mail.

Part I: Introduction to the IAS Process Model

2 Objective and Application of IAS Process Model

The IAS Process Model (IAS-PM) defines a uniform and mandatory procedure of activities and results for student projects and theses. 4 process model classes have been defined that describe separate rules for different thesis subject areas. These four process model classes are as follows:

- **Model for software development**
- **Model for system development**
- **Model for conceptional projects**
- **Open model**

In addition, the IAS Process Model defines different time requirements for Bachelor, Research, Study and Master Theses, by requiring different amounts of published documents.

While the first two process model classes mentioned have the common goal of developing a system with product characteristics, the third model focuses on creating a theoretical concept. The fourth process model class is generally open and only defines a general working procedure. It is intended for theses that cannot be categorized in one of the other three process model classes.

The actual creation of theses is accompanied by activities that deal with Quality Assurance (QA), Configuration Management (CM) and technical Project Management (PM).

2.1 The V-Model

The development of the IAS-PM was based on the V model. The V model is a generic process model that does not prescribe a specific organizational structure or time sequence. Yet all necessary activities and products, as well as their interdependencies, for project execution have been defined. First, the V model was split, which means the activities and products were adapted to the special needs of small single-person projects, while at the same time the organizational structure of the IAS was taken into consideration (organizational tailoring). In order to define the time sequence of project execution for theses the V model was combined with a simple phase model.

2.2 Objectives of the IAS Process Model

The standardized process model offers support in reaching the following goals:

- Educating students in the areas:
  - Software technology
  - Project management and project execution
  - Teamwork
  - Documentation

- Improving and guaranteeing quality:
  - A standardized procedure is the best guarantee for complete results.
  - Pre-defined intermediate reports allow for early test measures.
  - Uniform product contents simplify the readability of products and the execution of test measures.

- Re-usability:
  - Standardized procedures allow universal solution approaches to be recognized which may be re-used.
  - (Intermediate) results are standardized in such a way that, if necessary, other people can understand them within a reasonable amount of time.

2.3 Fulfilling QA Rules

The process model contains detailed rules for the total project including the areas of project management, quality assurance and configuration management. The process model adopts the requirements made by ISO 9001.

3 Introduction to the Process Model

The process model defines all aspects of

- Activities and products

as well as

- Product states and logical dependencies between activities and products

during project execution.

The process model consists of four submodels (Figure 3-1). Besides the submodel Project Execution (PE), there are the submodels Quality Assurance (QA), Configuration Management (CM) and Project Management (PM). The quality assurance tasks encompass the definition of requirements like test methods and test criteria and all product testing. Results are reported to project management. The goal of configuration management is to create a product, e.g., a document, that is uniquely identifiable in its functional and exterior characteristics. This identification provides for a systematic control of changes and secures integrity. Project management plans, checks and controls the project's internal activities. It is also the interface to project external units like other projects. The process model controls the interaction between individual submodels. This allows for an allocation of various tasks to different persons or teams and also guarantees a smooth procedure.

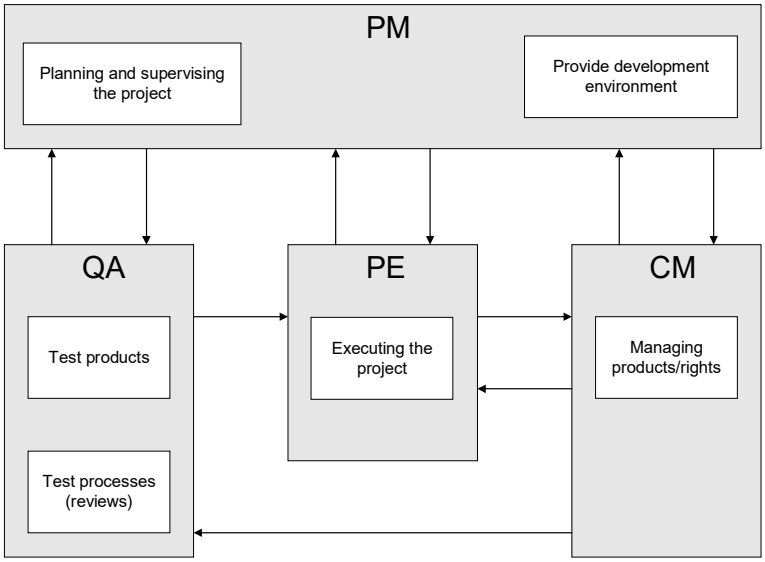

Figure 3-1: The four submodels PE, QA, CM and PM

The submodels Project Management (PM), Configuration Management (CM) and Quality Assurance (QA), which describe accompanying activities in a development project, are the same for all process model classes. Therefore, they are described in the following. In contrast, the submodel Project Execution (PE) is gravely different for each process model class. For this reason, the submodel PE is described separately for each process model class in the second part of this process manual.

For better differentiation, the submodel PE has been given separate names for the individual process model classes. They are:

- Software Development (SWD) for the model for software development projects
- System Development (SD) for the model for system development projects
- Concept Development (CD) for the model for conceptional projects.

For the process model class Open Model, no separate name was given to the submodel PE.

3.1 The Organizational Structure

Figure 3-2 shows the organizational structure of the IAS-PM. A software project (software development model) has been used here as an example.

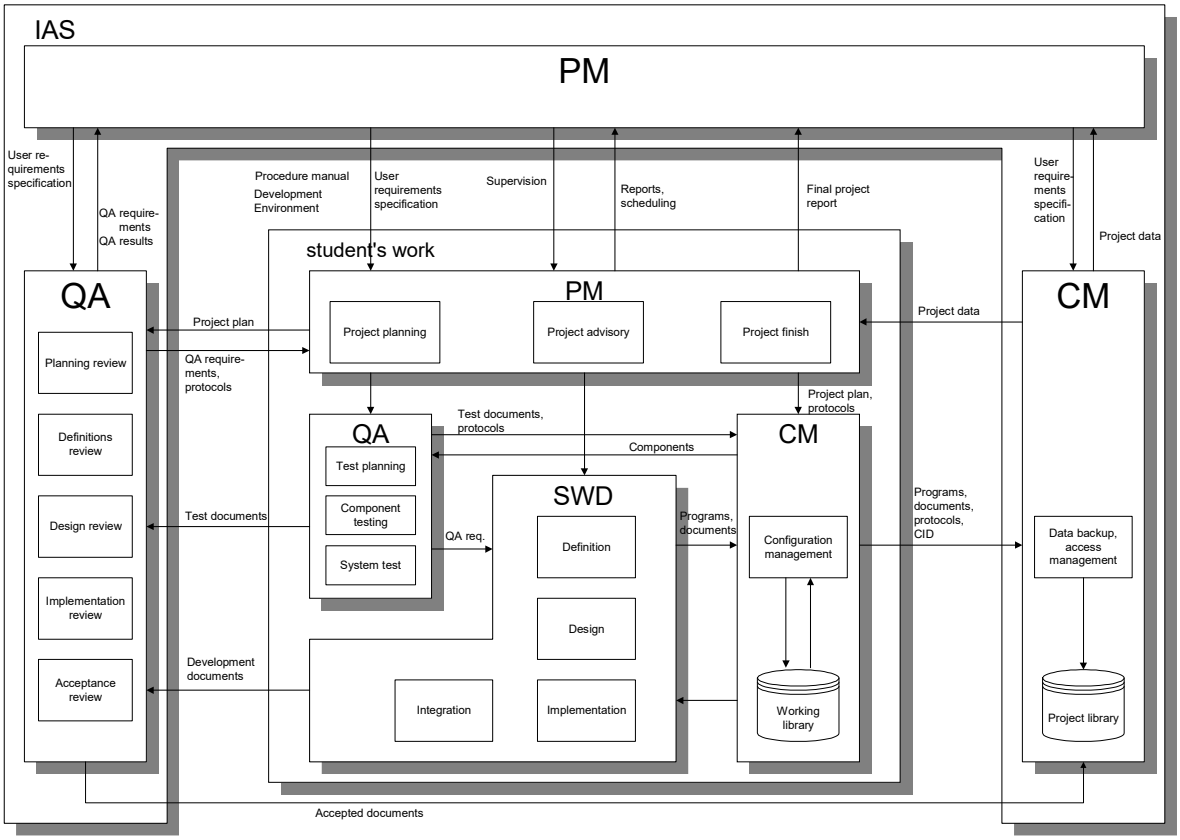

Figure 3-2: Organizational structure of the IAS-PM

A thesis or any other type of student work is executed as a project. The four submodels of the process model are embedded in the organizational structure of the IAS. For the purpose of education in the area of software engineering, the student will fulfill exercises in all four submodels. The IAS will support the student in the submodels Project Management (PM), Configuration Management (CM) and Quality Assurance (QA). In the submodel software development (SWD) the student is solely responsible for the complete execution of his task. The student's project might be integrated in a larger project. In the framework of the student's project an intermediate task of the major project will also be fulfilled.

3.2 Structure and Content of the Process Model

The basic elements of the process model are the activities and products that are executed and processed during the project execution process.

3.2.1 Activities and Products

An *activity* is an action which can be exactly described in reference to its results and execution. Activities can consist of a row of defined "intermediate activities", if each of these intermediate activities has its own defined intermediate results.

A *product* is either the processed object or the result of an activity. Analog to the subdivision of activities in intermediate activities, products may also be subdivided into "intermediate results" (e.g., individual chapters of a document).
An activity can be

- the creation of a product,
- a change in the state of a product or
- the change of product contents.

These basic elements "activity" and "product" are represented by special symbols (see Figure 3-3).

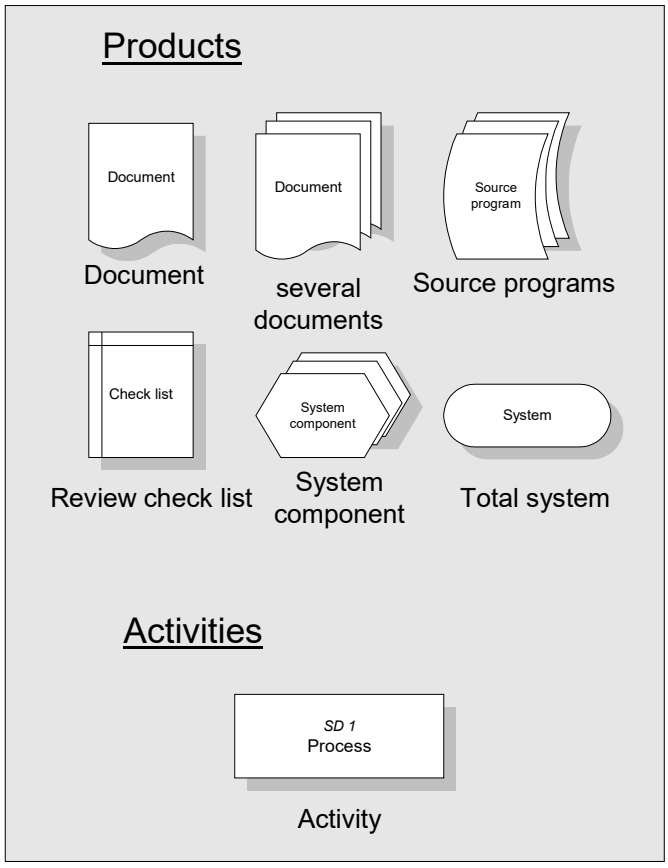

Figure 3-3: Symbols for products and activities

For each activity there must be an activity description in the form of an operation manual, which is to be adhered to during the execution of the activity. If an activity is subdivided into intermediate activities and logical dependencies between intermediate activities and products must be illustrated, then the activity schematics must contain an activity subdivision.

Each *product* must have a product description which defines the contents of the product. The product description occurs according to a fixed pattern, the *product scheme*. For documents theses schemes are provided in the form of document templates (Word).

3.2.2 Product States

When processing (intermediate) products in the process model, please note that certain products go through different *states* during the development process. Products may assume the following states:

"in progress"
The product is being worked on. Either it is in the “private” development area of the developer or it is under the control of the developer within the project library.

"submitted"
From the perspective of the creator the product is finished and has been stored under *Configuration Management*. A Quality Assurance test is performed on the product. If the product is denied by QA then it will return to the “in progress” state, otherwise it will move forward to the “accepted” state. After the “submitted” state the creator can only perform modifications to the product using sequential version numbers.

"accepted"
The product has been tested and released by QA and may only be changed within a new version.

The states and their legal state transitions are shown in Figure 3-4.

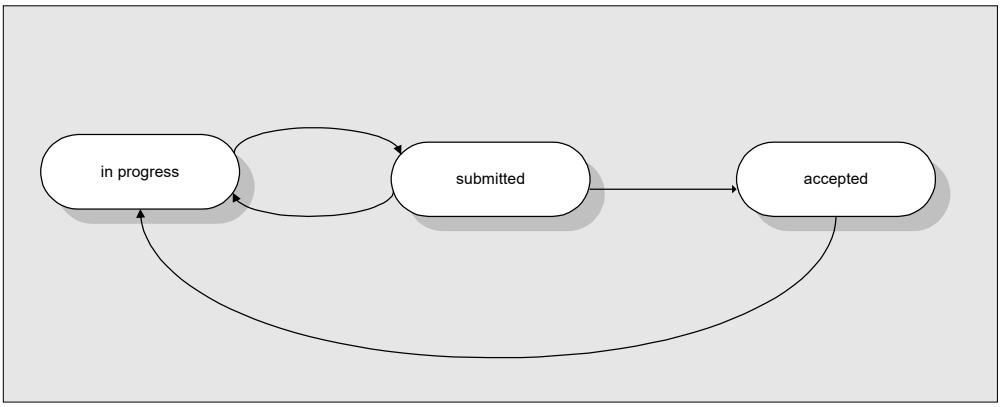

Figure 3-4: States and state transitions of products

3.3 Overview of the IAS Process Model

In the following an overview of the activities and products as well as the responsible persons involved is illustrated. A software project, that was regulated by the software development model, will be used as an example.

The activities are assigned to persons with the following abbreviations:

ID:	Institute Director
A:	Scientific Staff Members (supervisor or co-supervisor)
S:	Student
DBM:	Data Backup Manager

Activities and products that are indicated with these parentheses [] are optional.

Software Development (SWD)

SWD 1 Definition Phase (S)

SWD 1.1	Determine and define requirements	→	System requirements specification
SWD 1.2	Analyze requirements Define user interface	→	Software system model

→ Definition review (A, S)

SWD 2 Design Phase (S)

SWD 2.1	Design software system architecture	→	Software system architecture
SWD 2.2	Design and specify software components	→	Specification of the software components

→ Design review (A, S)

SWD 3 Implementation Phase (S) → Source programs with integrated documentation

→ Implementation review (A, S)

SWD 4 Integration and Acceptance Phase (S)

SWD 4.1	Integrate and install system	→	Installed system
SWD 4.2	Create user manual	→	User manual

→ Acceptance review (A, S)

Quality Assurance (QA)

QA 1 Process Test (ID, A, S)

QA 1.1	Definition review (A, S)	→	Definition review protocol
QA 1.2	Design review (A, S)	→	Design review protocol
QA 1.3	Implementation review (A, S)	→	Implementation review protocol
QA 1.4	Acceptance review (A, S)	→	Acceptance protocol

QA 2 Component and System Test (S)

QA 2.1	Create test specification	→	Test specification
QA 2.2	Test system components	→	Test protocol (components)
QA 2.3	Test total installed system	→	Test protocol (system)

Configuration Management (CM)

CM 1 Configuration Management (S) → CID (Configuration Identification Document)

CM 2 Data Backup (S, DBM) → USB sticks

Project Management (PM)

PM 1	**Planning Phase (A)**	→ User requirements specification
PM 2	**Project Planning (S)**	→ Project plan
PM 3	**Project Advisory (A, S)**	→ Review protocols (from QA 1)
PM 4	**Project Finish (S)**	→ Final project report

General Documents

→ Literature
→ Terminology
→ Abbreviations

Responsibilities

Figure 3-5 shows the responsibilities for the creation of the individual projects during the development process on the example of a software project.

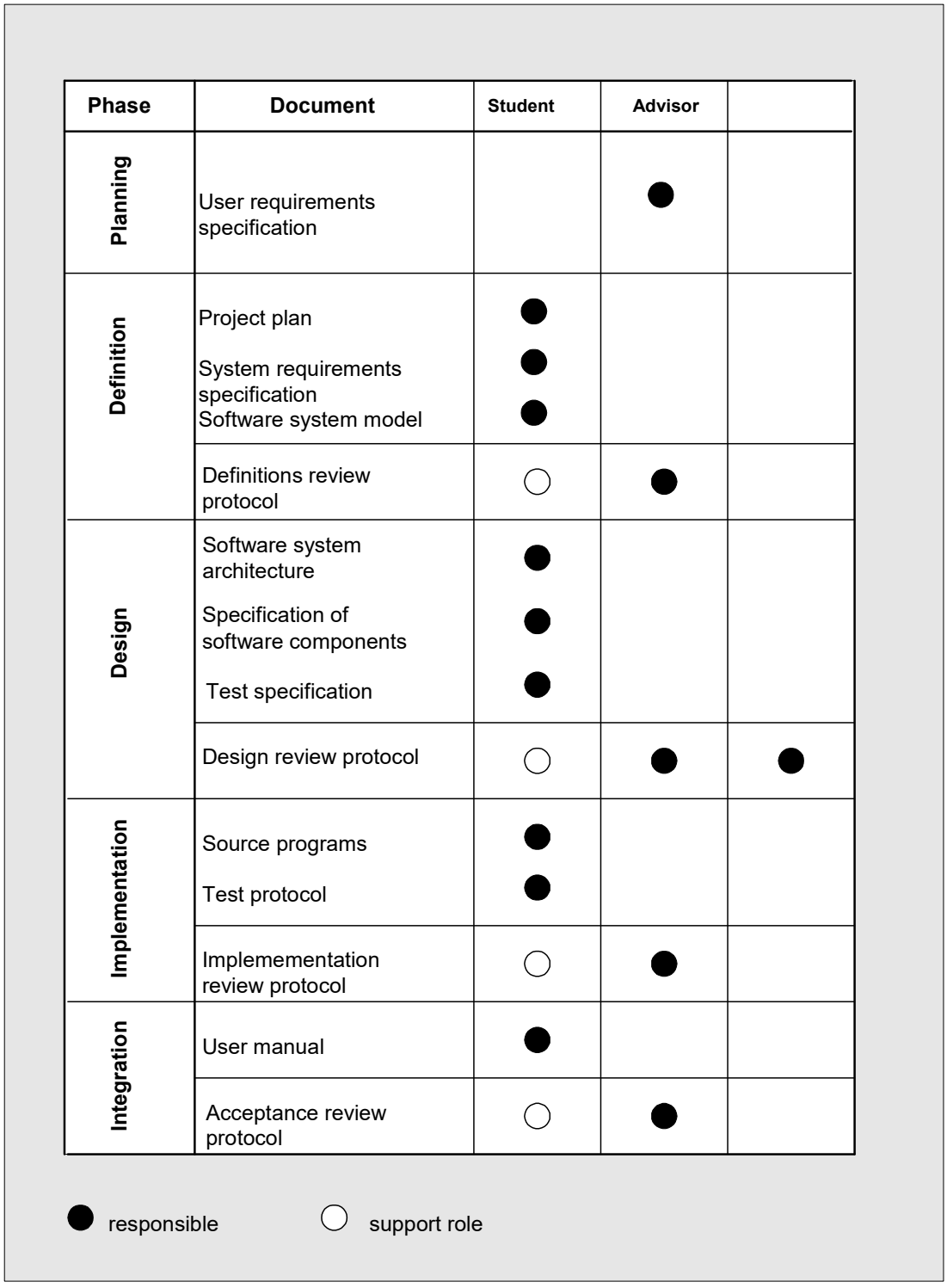

Figure 3-5: Responsibilities for individual products

4 Submodel Configuration Management (CM)

The goal of the configuration management (CM) is to provide for a unique identification of a product e.g., a document, in its functional and external characteristics. This identification serves as a systematic control of changes and guarantees integrity.

The configuration management supervises the configurations during the complete development so that relationships and differences between earlier configurations and the current configuration can be recognized immediately. The CM also provides for immediate access to earlier versions. Therefore, changes are traceable and can be tested.

In the frame of configuration management products (documents and code) are initialized and recorded. When dealing with products please note that products pass through various states during the development process. Possible states for a product have been defined in Section 3.2.2. Figure 4-1 shows how the version number changes during state transitions. The transition from one state to another is triggered by an activity. As a general rule a transition of state also means a change in access rights.

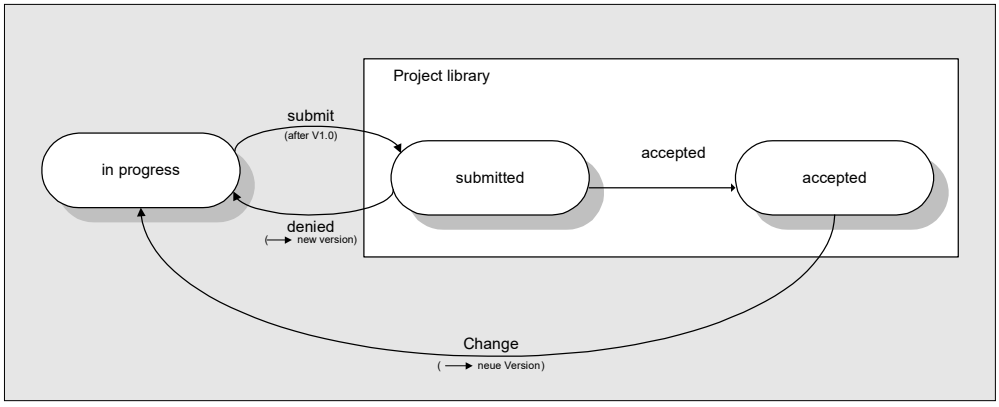

Figure 4-1: State transitions and versions

Figure 4-2 shows an overview of the submodel configuration management.

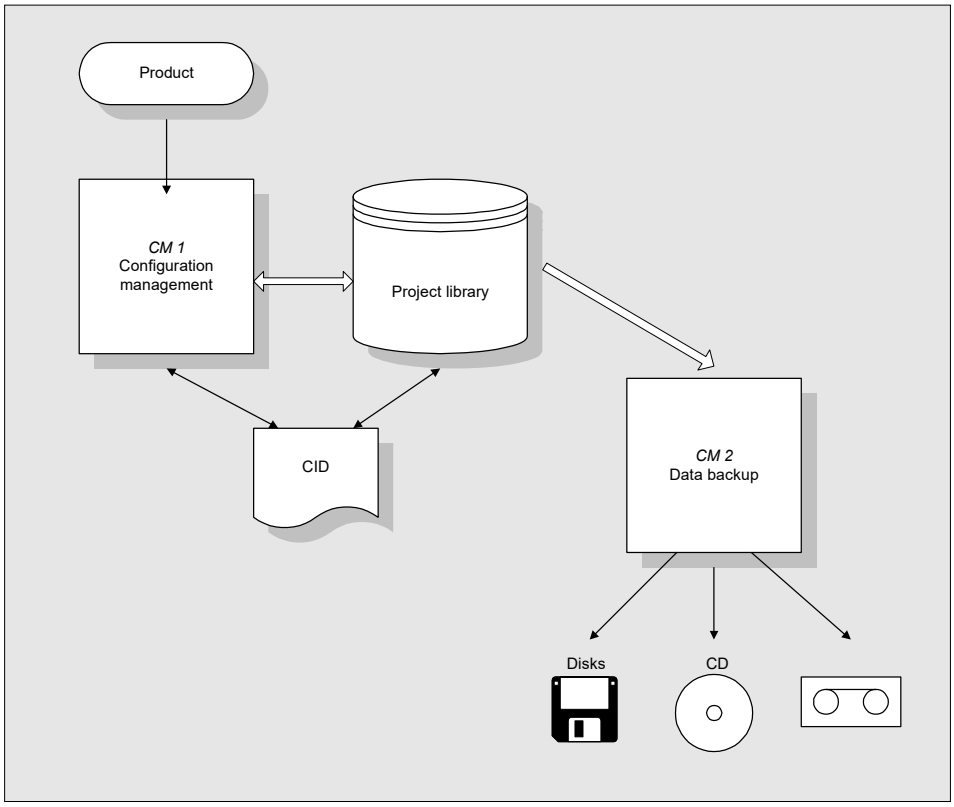

Figure 4-2: Overview of the submodel configuration management (CM)

4.1 Configuration Management (CM 1)

All objects of a configuration (documents, protocols, source code files, etc.) are archived and catalogued by the configuration management, so that objects cannot be destroyed by accident or on purpose. All objects are uniquely identifiable. Therefore, changes can be followed during the project execution process and during normal use, so that precise intermediate insertion points for further changes are provided.

A project directory is created at the IAS for each project upon project begin. The supervisors and the data backup manager have “write” authority for this directory. Documents are placed into this directory as soon as they have received the status *submitted*, which means they have been given to the supervisor. After that they are subject to configuration management.

4.2 Document Version Management

In the frame of the configuration management each product has the CM-required information for identification. Automated templates support the updates and version management of documents.

The data on the cover sheet of the document (see **Fehler! Verweisquelle konnte nicht gefunden werden.**), in the header and footer and in the document version management table (see Figure 4-) need to be entered manually. The table of contens is updated automatically via field functions and should not be changed manually. This simplifies document handling and secures data consistency.

When originally creating a document, it will receive the “in progress” state first and the version number 0.1. Documents in the “in progress” state can be changed, stored and printed at will. Important changes though should be documented. The first finished version, from the perspective of the student, will receive the version number 1.0. After version 1.0 storing the document is only allowed in the “submitted” state. Documents with a lower version number can not be stored in the “submitted” state.

After receiving the “submitted” status the documents are subject to the configuration management and should not be overwritten anymore. After any change the document must be stored under a new version number with a record of the changes made. The version numbers need to be incremented by 0.1 and the file names need to be entered according to the naming convention.

Procedure for creating a document:

1) Open the template. When you open the template the general project data should be entered by means of a dialog form.
2) Read the hints (*in italics*) and delete them.
3) Work on the document.
4) Store the document under a file name according to the naming convention.

Document Version Management

Version	Author	QS	Date	Status	Changes
0.1	Sample	Ri	01.12.00	In progress	Creation
1.0	Sample	Ri	01.12.00	submitted	
1.0	Sample	Ri	01.12.00	accepted	

Figure 4-4: Document version management

4.3 Version Control of Source Code

If source code is created during the project, it must be put under version control. For that purpose we use GIT, which is freely available, at our institute. As client TortoiseGIT is provided for all students. Source code files are stored in the repository, a kind of database. GIT enables the possibility to set up a central server, where all repositories are administrated. Moreover, local repositories can be set up in a home directory. If a central or a local repository should be used, has to be adjusted with the supervisor. A separate document and plenty of examples in the internet are available, which explains the necessary steps and the usage of GIT.

4.4 Data Backup (CM 2)

At regular intervals all data present under the CM will be stored. This continuous data backup is performed automatically.

Before a project is finished all project-relevant data are securely stored so that they are reproducible at any time and a later reuse of individual data or the whole project is possible.

5 Submodel Project Management (PM)

The Project management (PM) has the following tasks:

- The PM *plans*, *guides* and *supervises* the project's internal activities.
- The PM is the *interface* to the project's external units e.g., other projects.
- The PM composes the *information center* of the project.

Figure 5-1 shows an overview of the submodel project management.

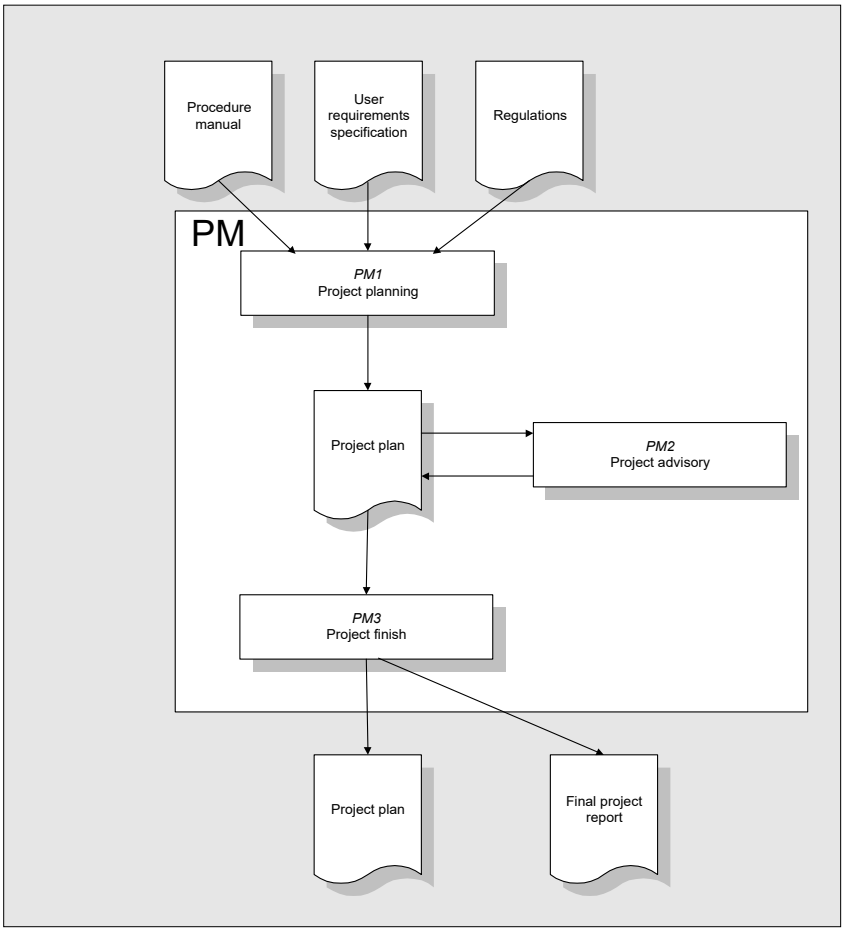

Figure 5-1: Overview of the submodel project management (PM)

5.1 Project Planning by the Supervisor

In the planning phase the basic requirements of the system to be developed are determined. The planning phase includes various studies in the preliminary phase of a project. It also includes an operability study and an effort and schedule estimate. Because the time frame for an advanced project or thesis project is prescribed, the requirements must be defined in such a way that the project can be completed within the expected time frame. If necessary, then the requirements must be adjusted. The basic requirements are recorded in the user requirements specification. The planning phase must be completed before the student's project may begin.

5.1.1 User Requirements Specification

5.1.1.1 Purpose of the Document

The user requirements specification is a summary of all basic academic requirements that the proposed system must fulfill. The basic requirements represent a concentration of the fundamental characteristics of a system and their description on an abstract level. The user requirements specification is the first product that describes the requirements of a new system in a general manner.

In the section *Execution* the work sequence is defined. This means the supervisor determines which process model class is to be applied to the project, which project phases must be completed, and which documents are to be created.

5.1.1.2 Structure Scheme

0 Table of Contents

1 Introduction to the Project

2 Objectives

3 Operational Area

4 Functional Requirements

5 Non-functional Requirements

6 Quality Requirements

7 Execution

8 Additions

9 Literature

Amount: ca. 3-6 pages

5.2 Project Planning by the Student (PM 1)

Project planning includes the intermediate tasks organizational planing, effort and time planning as well as resource planning. The organizational planning defines the organizational structure of the project as well as the process organization based on the project structure plan. Especially the persons involved in the project and their roles are to be named. Interfaces to external persons that are important for the project are also to be named. Usually an effort and resource plan is not necessary for advanced undergraduate projects and master theses. A schedule in the form of a Gantt chart is very important. Project planning is documented in the *project plan* and is to be followed.

5.2.1 Project Plan

5.2.1.1 Purpose of the Document

The project plan consists of planning and defining project organization, project sequence, time and resource amounts for all four submodels. It is an instrument used by project management to plan, guide and supervise. The project plan consists of a description of the individual packages, a project structure plan and two bar charts (Gantt chart). The project structure plan is a graphical representation of the project organization with individual project packages. The bar chart shows the time sequence of the project whereas the original plan and the actual project sequence upon project finish are differentiated.

Note:
Advanced undergraduate and master thesis projects should include a sufficient buffer time for unforeseen delays. A three-month project should have a buffer time of two weeks, a six-month project should include four weeks' buffer time.

5.2.1.2 Structure Scheme

0 Table of Contents

1 Project Organization

2 Project Structure Plan

3 Work Packages

4 Milestones

5 Bar Chart (planned)

6 Bar Chart (actual)

Amount: ca. 7-8 pages

5.3 Project Advisory (PM 2)

The student is solely responsible for a smooth and timely project sequence. Some of these responsibilities are, for example, regular reports to the supervisor, punctual scheduling for reviews, timely submission of documents, fulfilling deadlines.

The supervisor, in his role as a superior project manager, carries the responsibility for a smooth and timely project sequence and to supervise the student. In order to guarantee a smooth project sequence, it is necessary to provide required resources (e.g., computers) punctually and to introduce the student to the subject and the development environment.

The IAS process model promotes teamwork by prescribing project group meetings on a bi-weekly basis. Students report on their progress in these meetings and reviews are performed at pre-determined milestones.

5.4 Project Finish (PM 3)

A final project report is to be made by project management that contains a summary of the project sequence and the accomplished goals. Further a short report containing the experience and knowledge gained during the development process is to be completed that can be used in follow-up projects.

5.4.1 Final Project Report

5.4.1.1 Purpose of the document

This document summarizes the process of the project. Experiences and problems are evaluated and analyzed to improve follow-up projects.

5.4.1.2 Structure Scheme

0 Table of Contents

1 Summary

2 Experiences

3 Problems

Amount: ca. 3-4 pages

Hint:
At the end of any advanced undergraduate or thesis project the student must write a thesis paper in addition to this document. Of course, the results from the process model may be used, as long as the contents were created by the student. The paper is to be written as a conclusive text. The structure is not prescribed.

6 Submodel Quality Assurance (QA)

Quality is defined as “the characteristic of a unit with regard to its suitability to meet defined and expected requirements" (/DIN 55350/).

The basic requirements that are applied to the propose system development are refined during the project work and are recorded in the products. The measures described in the submodel Quality Assurance (QA) guarantee that the prescribed requirements are fulfilled.

Software quality is assured by using constructive quality assurance measures and analytical quality assurance measures.

Constructive Quality Assurance Measures

Quality cannot be retrofitted into a product. Therefore, it is necessary to assure quality through constructive measures. The objective is to prevent quality flaws and to construct testable products.

Constructive quality assurance measures can be e.g.,

- structuring the development process with a process model
- supporting the development process with methods and tools

Constructive quality assurance measures are defined in the submodel QA. The application of constructive measures occurs in the submodel Project Execution (PE).

Analytical Quality Assurance Measures

The goal of analytical quality assurance measures is to test and evaluate the quality of the test objects. They are defined and applied in the submodel QA. Analytical quality assurance measures are applied to the products and activities of all submodels.

6.1 Process Testing (QA 1)

Process testing is a spot test of an activity to see if it is in keeping with agreed rules. Such rules can be:

- Documentation and coding standards
- Application of certain methods and tools
- Procedures for data backup, version management, etc.

Process testing refers to all activities of the individual submodels. During a process test selected activities are evaluated based on their fulfillment of prescribed procedures and project standards. The documents generated are especially checked. Process testing is done during planned *reviews*.

The goal of a review is to identify errors and to search for breaches against specifications, standards and plans. Reviews utilize the human cognitive and analytical capabilities to evaluate and test complex issues. A review consists of the following steps:

- Review planning
- Review preparation
- Review meeting

The student is responsible for scheduling reviews during project planning. Upon completion of a phase in the process model, the supervisor and the student undertake the review. The results of the individual tests are recorded in the review protocol (see Figure 6-1).

University of Stuttgart
Institute of Industrial Automation
and Software Engineering

Review (Software Development Project)

Purpose: Overview on project status, timeliness and delivery into sadapro-directory

Name:	**Type and Number:**

Phase 1: Definition Phase

Documents and Products of Phase 1	Version	Date
User Requirements Specification		
System Requirements Specification		
Project Plan		
Software System Model		

Acceptance of Phase 1

Date	Examiner	Remarks	Accepted ☐
			Delivered to sadapro ☐

Phase 2: Design Phase

Documents and Products of Phase 2	Version	Date
Software System Architecture		
Software Components Specification		
Test Specification		

Acceptance of Phase 2

Date	Examiner	Remarks	Accepted ☐
			Delivered to sadapro ☐

Phase 3: Implementation Phase

Documents and Products of Phase 3	Version	Date
Source Code (with integrated documentation)		
Test Protocol (Components)		

Acceptance of Phase 3

Date	Examiner	Remarks	Accepted ☐
			Delivered to sadapro ☐

Phase 4: Integration and Acceptance Phase

Documents and Products of Phase 4	Version	Date
Installed Software System		
User Manual		
Test Protocol (System)		

Figure 6-1: Review-Protokoll

Whenever a student submits a product (e.g. one of the required documents), the supervisor checks it. The checking ends with the determination of the results. The following results are possible:

- A product is accepted, if necessary, with slight improvements required, without a further test being necessary.

- A product is rejected (or not accepted) and must be re-submitted after improvements have been made.

The student must implement the required changes and submit all required documents for this phase before the review. During the review, the respective products and version numbers are inserted for each item. Additionally, the supervisor compares the current state of the project to the project schedule.

In the following the individual reviews are described. They are the same for the two process model classes *model for software development* and *model for system development* and are printed in this part of the manual. In the second part of the process manual the reviews for the process model class *model for conceptional work* have been separately printed because they differ from those for the other two classes. Reviews have been prescribed for the *open model,* but no product processes are described in the following.

6.1.1 Definition Review (QA 1.2)

The documents *project plan*, *system requirements specification* and *system model* are checked in the definitions review. Figure 6-2 shows the product flow chart of a definitions review.

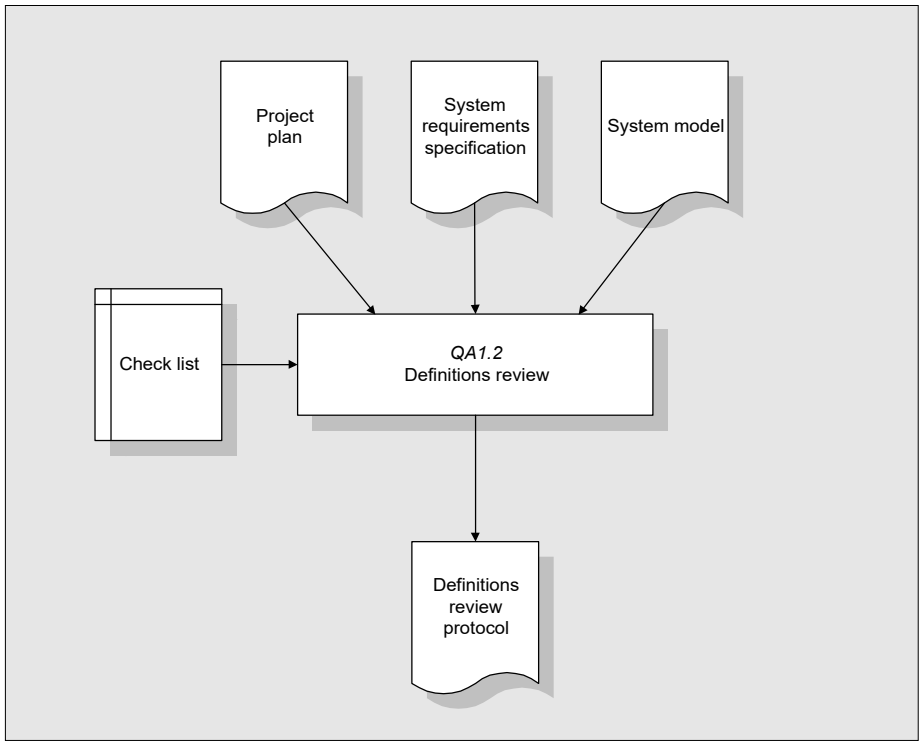

Figure 6-2: Product flow chart for the definitions review

6.1.2 Design Review (QA 1.3)

The *system architecture*, the *specification of system components* and the *test specification* are checked in the design review. Additionally, improvements stemming from the definitions review can be made. Figure 6-3 shows the product flow chart of the design review.

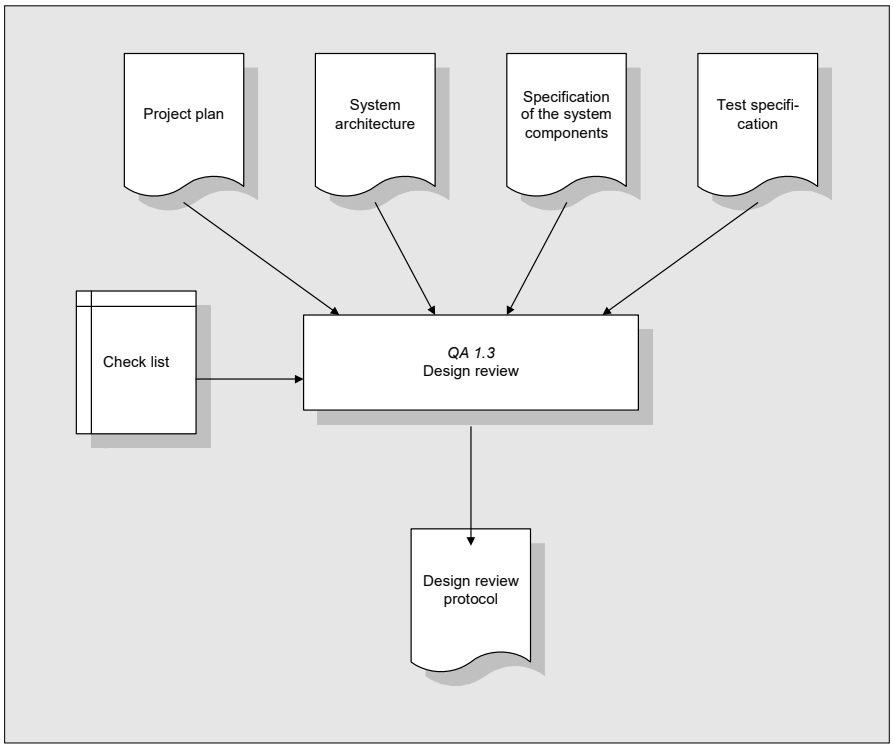

Figure 6-3: Product flow chart of the design review

6.1.3 Implementation Review (QA 1.4)

During the implementation review the generated *source programs* are checked in a code review and adherence to programming regulations is confirmed. Additionally, improvements stemming from the design review are completed. Figure 6-4 shows the product flow chart of an implementation review.

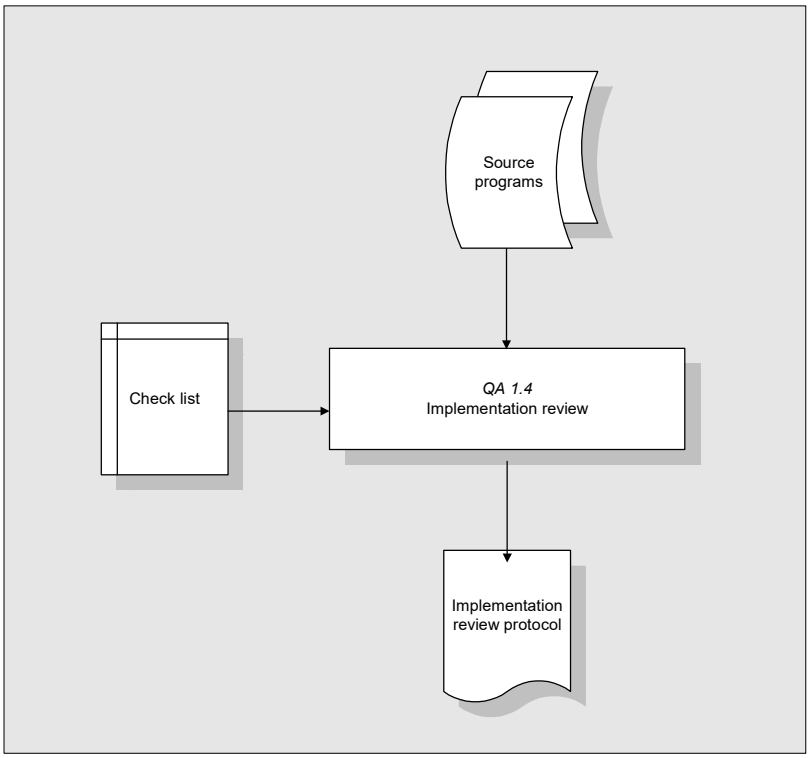

Figure 6-4: Product flow chart of an implementation review

The code reviews are performed in the middle and at the end of the implementation phase (i.e. two code reviews per student). In doing so, problems of programming can be recognized in time. Code reviews take place in student projects meetings. They should be performed in the following way:

- 3-4 days before a meeting a code section should be determined (scale: approx. 3-4 pages). All students of the teams have to look over this code section in these days. They must check if the programming regulations are kept.
- The results have to be discussed in the student project meetings.

For all projects, in which no program code is created, the supervisor has to decide, which kind of review should be done (e.g. model review). In each project at least one review has to be performed in the students' project meeting!

6.1.4 Realization Review (QA 1.4)

The realization review checks the hardware components on optical cleanliness as well as adherence to common regulations. Additionally, improvements stemming from the design review may be completed. Figure 6-5 shows the product flow chart of a realization review.

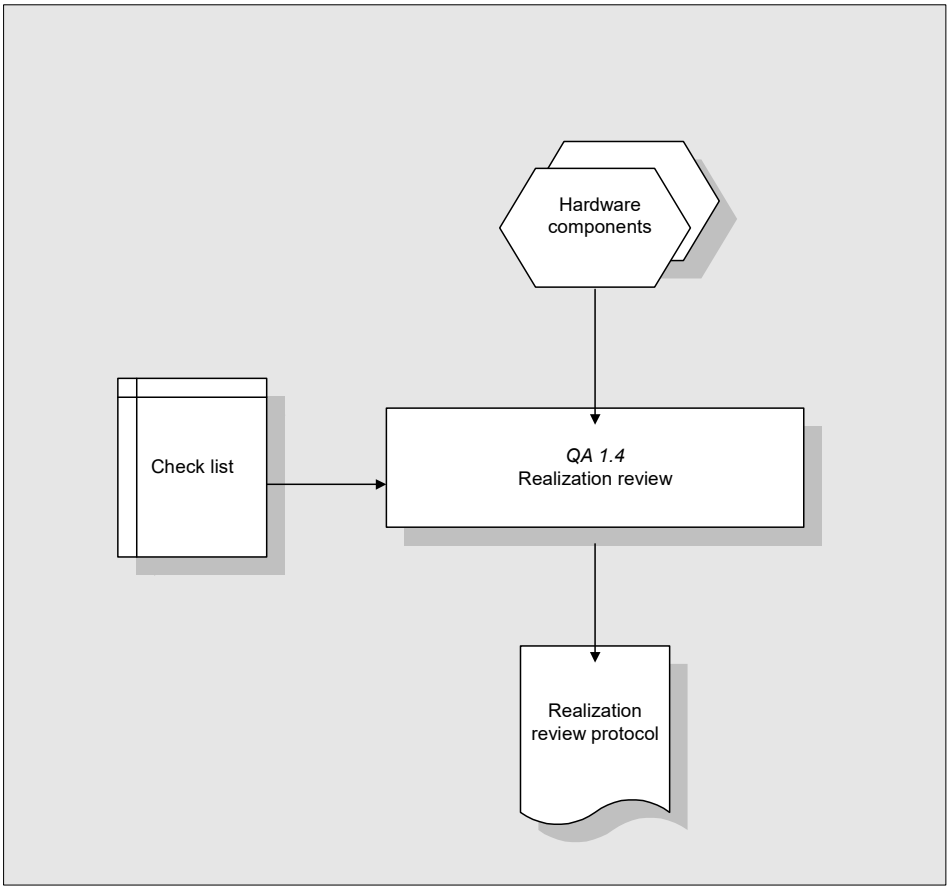

Figure 6-5: Product flow chart of a realization review

6.1.5 Acceptance Review (QA 1.5)

When the system is to be accepted, its functionality and operability are tested by the contractor and the supervisor. The results are recorded in the *acceptance review protocol*. Figure 6-6 shows the product flow chart of the acceptance review.

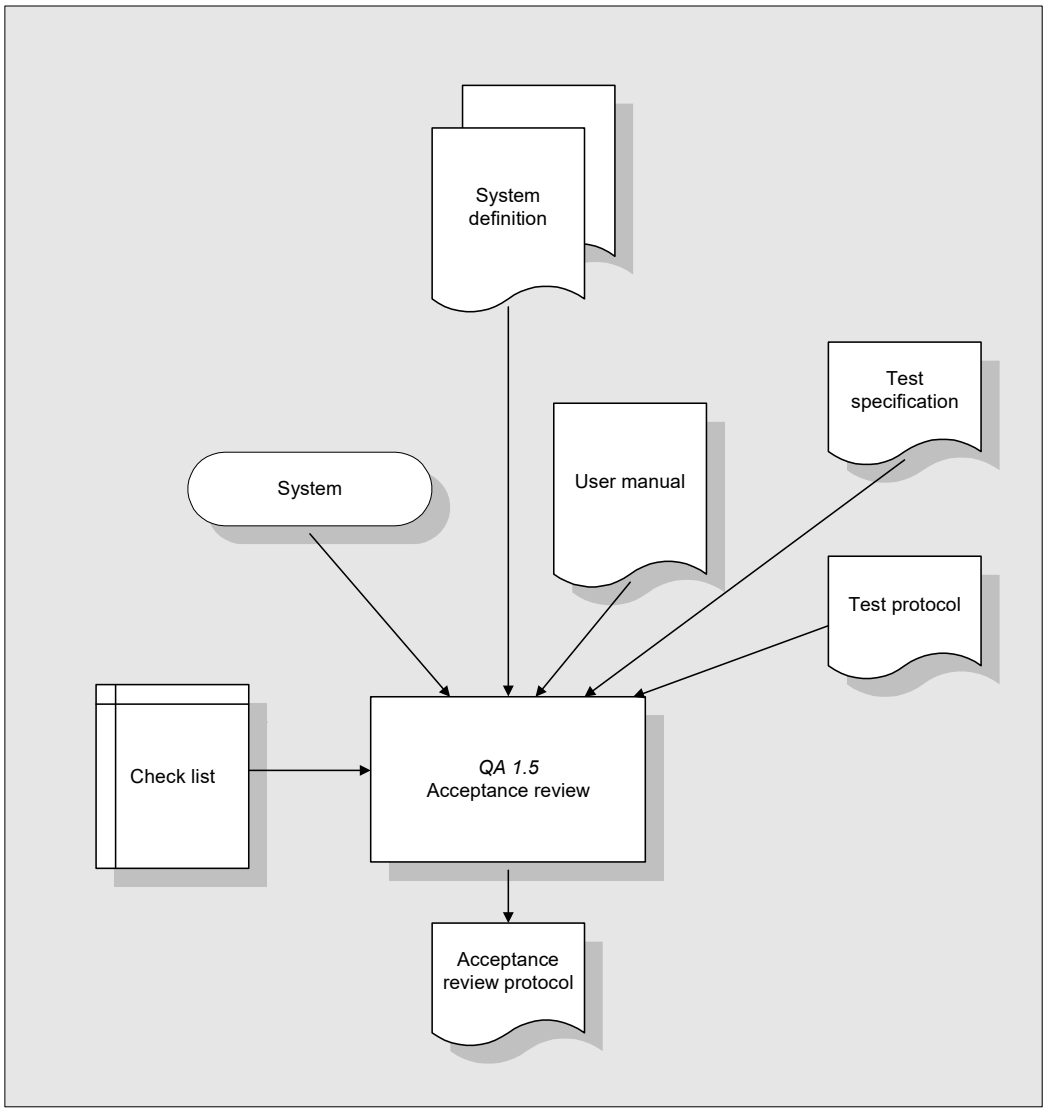

Figure 6-6: Product flow chart of the acceptance review

In order to guarantee that the developed product can be installed and used, the acceptance review should be done in the following way:

- The student submits the product and the corresponding user manual to his supervisor, before the review. The supervisor stores the files in the Sadapro directory.
- It must be checked if all data are available for start-up in the Sadapro directory.
- The supervisor must put the product in operation. Only the information of the Sadapro directory can be used.
- Only if the supervisor has successfully put the product into operation, the acceptance review is complete.

6.2 Component and System Test (QA 2)

The component and system test serve to prove that the developed system components and the system as a whole fulfill the specified requirements and the expectancy of the contractor. Proof of evidence can be accomplished by validation or verification. Figure 6-7 shows an overview of the component and system test.

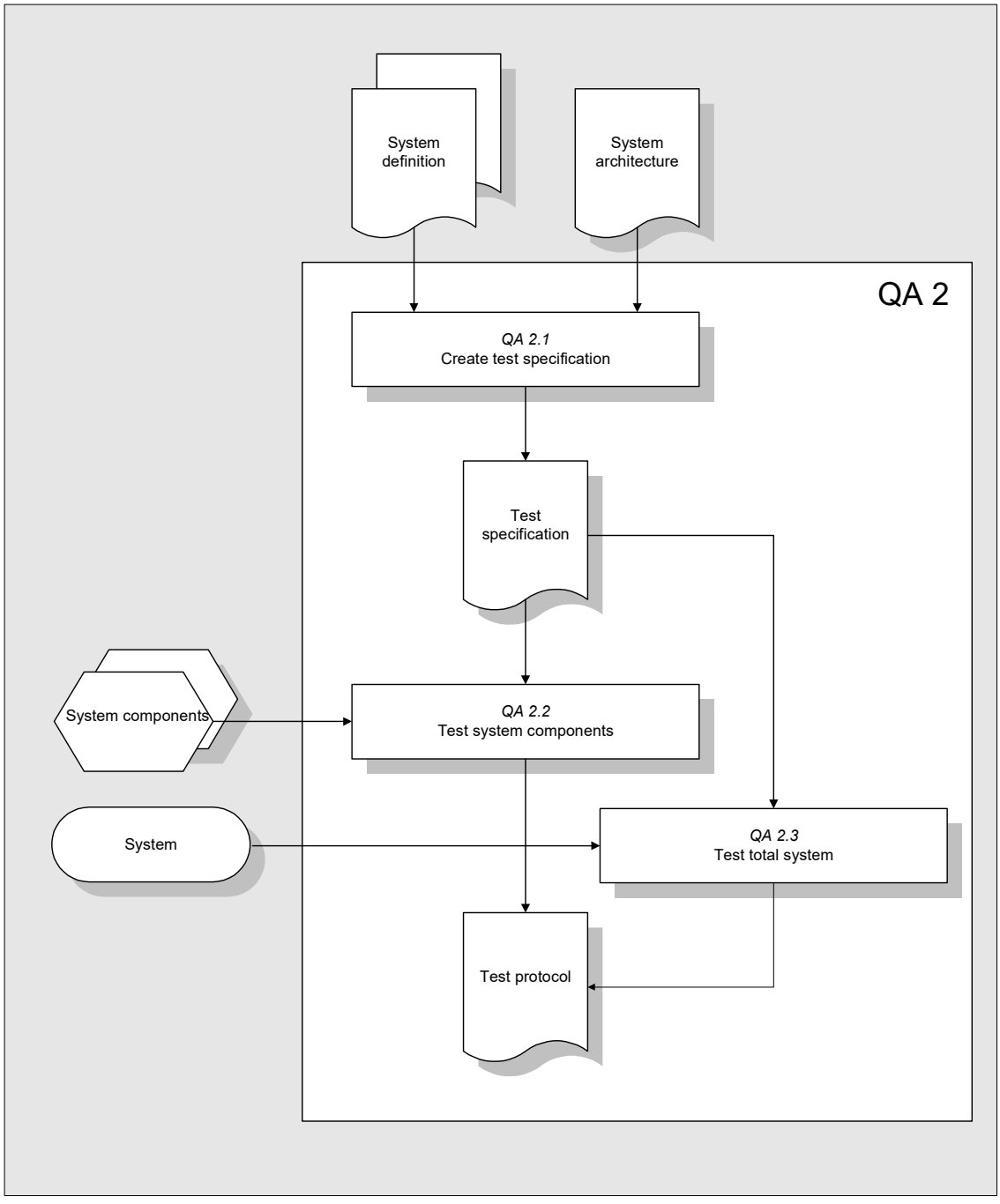

Figure 6-7: Overview of the component and system test (QA 2)

The principle of test is to execute the test object in defined test cases and to prove that the test object contains no errors for the defined test cases. The following criteria are to be fulfilled: The test must have an unambiguous specification reference and it must be reproducible, traceable and measurable. Test execution occurs in the following steps:

- Test preparation
- Test execution
- Test evaluation

Test preparation includes preparing the test cases, creating a test frame environment and preparing the recording of test results.

A special part of the test evaluation is comparing expected and actual test results. Specific analyses can be applied for evaluation needs.

Testing system components is to be done according to blackbox procedure, unless otherwise specified. The objective of the blackbox test case design is to discover situations in which the test object does not behave according to the requirements or specifications.

The blackbox test case design is derived from the requirements or specifications. The test object is considered a black box i.e., the tester is not interested in the internal structure or behavior of the test object.

The following blackbox test case design methods can be differentiated:

- Equivalent class building
- Limit value analyses
- Intuitive test case generation
- Functional coverage

6.2.1 Creating Test Specifications (QA 2.1)

General test requirements are derived from the requirements posed to the developed system. After that test methods and criteria are defined, which are used to meet test requirements.

Test cases are to be defined for individual components and for the system as a whole. Due to time limits test cases for advanced undergraduate projects and theses do not need to be exhaustive. It is sufficient to define a few test cases (e.g., for especially critical components, functions). In any case the global test cases named in the system requirements specification must be recorded and specified in more detail.

The test specification must be submitted before the implementation phase, or the realization phase respectively i.e., at the latest before the design review.

6.2.2 Test Specification

6.2.2.1 Purpose of the Document

The test specification includes a description of test requirements a goal, test methods and test criteria derived from the requirements as well as test cases. A coverage matrix will document the covering of all test cases. With the help of the test specification a decision can be made if the test was successful or not.

6.2.2.2 Structure Scheme

0 Content

1 Test Requirements

2 Test Methods

3 Test Criteria

4 Test Cases

Amount: ca. 4-8 pages

6.2.3 Testing System Components (QA 2.2)

The individual components are tested by the student by applying the test cases defined in the test specification. For this purpose, the corresponding test environments and procedures must be generated. The results and evaluations of the individual tests are to be documented in the test protocol.

The supervisor will randomly spot check a correct test execution.

6.2.4 Test Protocol

6.2.4.1 Purpose of Document

The test protocol contains the process of the individual tests as recorded by the tester, especially the comparison of expected and actual results

6.2.4.2 Structure Scheme

0 Table of Contents

1 <Test case1>*

1.1 Reference to the Test Specification
1.2 Procedure of Testing
1.3 Results
1.4 Discussion

Amount: ca. 4-8 pages

6.2.5 Testing a Complete Installed System (QA 2.3)

The total system is to be tested according to the specified test cases in the test specification. The results of the test are to be recorded in the test protocol. The test protocol is to be submitted during acceptance of the system. If large mistakes are discovered the system cannot be accepted.

Part II: Process Model Classes

The process model is subdivided in the four following process model classes:

- **Model for software development projects**
- **Model for system development projects**
- **Model for conceptional projects**
- **Open model**

The process model classes *model for software development projects* and *model for system development projects* regulate the procedures surrounding advanced undergraduate and thesis projects, that deal with the development of systems with product characteristics. The process model class *model for conceptional projects* is intended for projects that focus on the development of a theoretical concept. In this case the concept may include software, hardware or software/hardware systems. The evaluation of a concept can occur based on a prototypical realization. Such a prototype does not need to fulfill the high-quality standards as demanded by the first three process model classes.

The process model class *open model* is mostly open and regulates the general work sequence. It is intended for advanced undergraduate and thesis projects, which cannot be categorized in one of the other model classes. The supervisor has the capacity to specify more details of the project before the project begins.

In the following chapters the process model classes are described:

7 Model for Software Development Projects

7.1 Objective

This process model class defines the process of advanced undergraduate and thesis projects that focus on creating a software system. In contrast to the process model class *model for conceptional projects* the emphasis here is to create a high-quality product.

During software development the following four phases are passed:

- Definition phase (SWD 1)
- Design phase (SWD 2)
- Implementation phase (SWD 3)
- Integration and acceptance phase (SWD 4)

Table 7-1 provides an overview of activities and products represented in the different phases for Bachelor, Research, Study and Master Theses respectively. Documents marked with an „X“ are mandatory, those marked with an „O“ are optional, documents with a „-“ symbol do not need to be created.

Project phase			BT/RT/ST/MT
SWD 1	**Definition phase**		
SWD 1.1	Determine and define requirements	System requirements specification	X
SWD 1.2	Analyze requirements	Software system model	X
QA 1.2		→ Definitions review	X
SWD 2	**Design phase**		
SWD 2.1	Design software system architecture	Software system architecture	X
SWD 2.2	Design and specify software components	Specification of software components	X
QA 2.1	Create test specification	Test specification	X
QA 1.3		→ Design review	X
SWD 3	**Implementation phase**		
SWD 3.1	Implement software components	Source programs with integrated documentation	X
QA 2.2	Test software components	Test protocol (components)	X
QA 1.4		→ Implementation review	X
SWD 4	**Integration and acceptance phase**		
SWD 4.1	Integrate and install system	Installed software system	X
SWD 4.2	Create user manual	User manual	X
QA 2.3	Test complete installed system	Test protocol (system)	X
QA 1.5		→ Acceptance review	X

Table 7-1: Overview of the activities and products in the model for software development projects

Figure 7-1 gives an overview of the software development. The individual phases are described in the following sections.

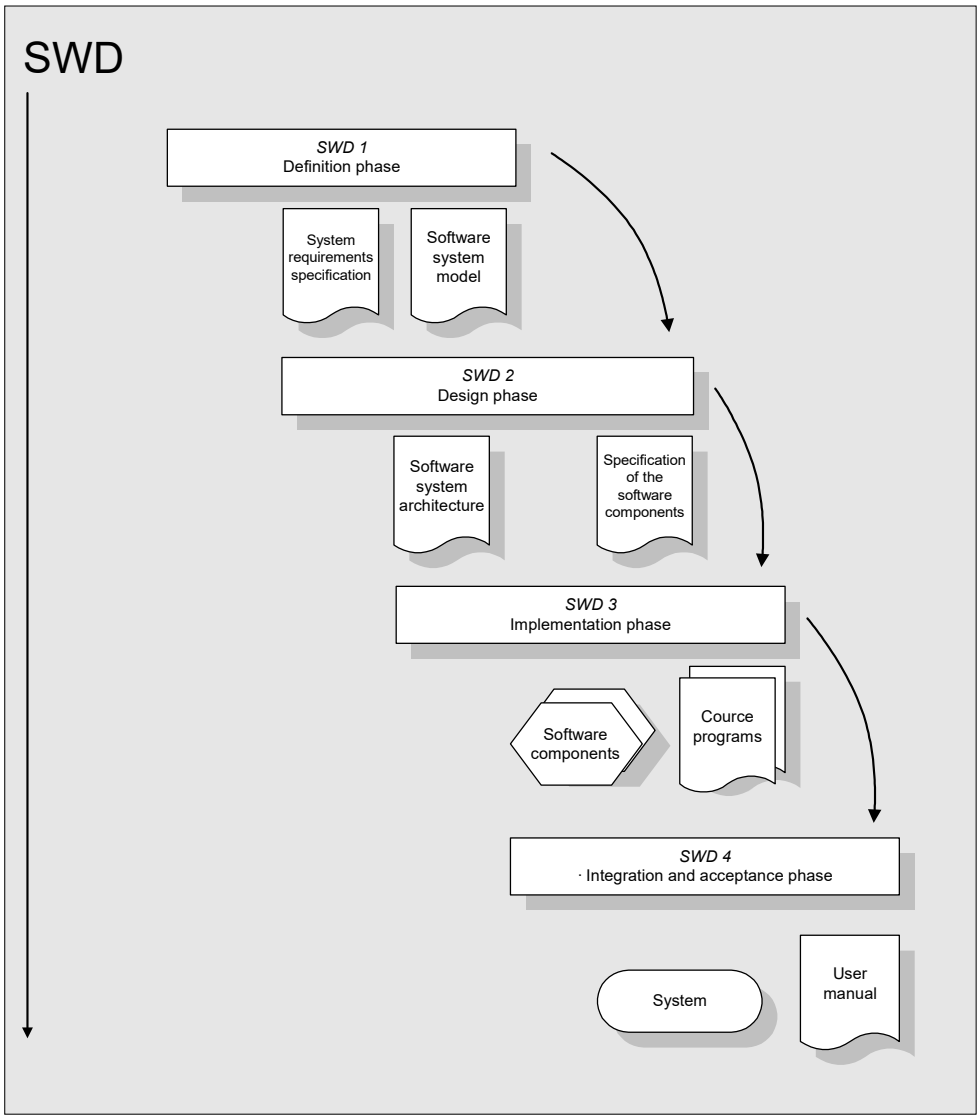

Figure 7-1: Phases and products in the model for software development

7.2 Definition phase (SWD 1)

In the definition phase the requirements are defined for the software system to be developed. The requirements determine the qualitative and quantitative characteristics of a system from the perspective of the contractor. The iterative process of determining the requirements is called system analysis. The *system definition* represents the fundament for the later acceptance of the system and is therefore very important. The system definition consists of the products *system requirements specification* and *software system model*.

The definition process is iterative. Normally several repetitions of the individual activities are necessary. The results of the definition phase are complete, unambiguous, consistent and operable requirements for the system to be developed.

In addition to the term definition phase the terms *analysis phase* and *specification phase* or *requirements engineering phase* are common.

Figure 7-2 shows the product flow chart for the definition phase.

Figure 7-2: Product flow chart for the definition phase

7.2.1 Determine and Define Requirements (SWD 1.1)

The basic requirements in the user requirements specification defined for the system need to be defined in a more precise manner. This is most often done by surveys and discussions with the supervisor and possible users. For an existing system, that is to be improved or replaced by the new system, a current state audit should be performed. This can often reveal weaknesses and flaws which can be taken into consideration when writing the new requirements.

After the requirements have been determined they need to be described and defined. For this purpose, they are classified and structured into several categories. Functional and non-functional requirements are differentiated. Depending on the type and complexity of the system a further subdivision may be necessary. In addition to functional and non-functional requirements further requirements for the user interface and quality must be defined. Also, development and target environments need to be determined. All requirements must be written down in the system requirements specification.

7.2.2 System Requirements Specification

7.2.2.1 Purpose of the Document

The system requirements specification is a summary of all academic requirements, that the system needs to fulfill from the perspective of the contractor. It contains the academic scope of the system in reference to its function, data, performance and quality. The system requirements specification defines *what* the system is to perform and not *how*.

7.2.2.2 Structure Scheme

0 Table of Contents

1 Objective

- 1.1 Mandatory Criteria
- 1.2 Optional Criteria
- 1.3 Limit Criteria

2 Operational Area

- 2.1 Application Areas
- 2.2 User Group
- 2.3 Operating Conditions

3 Environment

- 3.1 Software
- 3.2 Hardware
- 3.3 System Interfaces

4 Functional Requirements

5 Non-functional Requirements

6 Requirements for the User Interface

7 Quality Objective

8 Global Test Scenarios/Test Cases

9 Development Environment

- 9.1 Software
- 9.2 Hardware
- 9.3 Orgware
- 9.4 Development Interfaces

10 Execution

11 Additions

Amount: ca. 8-15 pages

7.2.3 Analyze Requirements (SWD 1.2)

Analyzing the requirements is often not enough to attain a sufficiently clear concept of the system to be developed. A system model forces an academic approach to the problem which is not possible if the requirements are stated verbally. Analysis occurs according to a suitable analysis method. We recommend using the analysis method *Use Cases*. In agreement with the supervisor another method may be used.

Use Cases

Use Cases describe how a system can be used to attain the target results. Use Cases treat the system like a blackbox. They are goal-oriented, and each Use Case describes a special way the system can be used. Because each system only needs to fulfill a limited number of tasks, there is only a limited number of Use Cases, that can describe the system.

The Use Cases are best determined by searching all external *agents* that interact with the system. When they interact with the system, they can assume different roles. These roles can be very different. An agent that performs a certain role is called an *actor*. Therefore, the search for Use Cases begins with finding all external agents and differentiating their roles.

A more detailed description of the application of Use Cases is published in the document template of the software system model.

Defining the Graphical User Interface (GUI)

For systems that involve a complex interaction with the user, a user interface (*GUI concept*) must be created. This concept describes how the system will present itself to the user. This is especially important when one considers the fact that the success or failure of a system often depends on its acceptance by the users. Prototyping is a good method for this activity.

7.2.4 Software System Model

7.2.4.1 Purpose of the Document

Creating a verbal system requirements specification is not sufficient, therefore a semi-formal or formal system model, the so-called software system model, must be created. The software system model describes the academic concept completely, unambiguously and consistently.

In case a software system with complex interaction between the user and the computer system is to be developed, the system model will describe how the system will present itself to the user and how it is to be used.

7.2.4.2 Structure Scheme

0 Table of Contents

1 Use Case Overview

2 Use Case Diagram

3 Description of Use Case <xxx>*

4 Description of Use Case <xxx>*

...

5 Sequence Diagram of Use Case <xxx>*

6 GUI Concept*

- 6.1 Principal Concepts of operation
- 6.2 Elements of the User Interface
- 6.3 Structure and Design of Windows
- 6.4 Design of Dialogs

Amount: ca. 6-10 pages

7.3 Design Phase (SWD 2)

In the design phase a system-technical solution in the form of a *software system architecture* is developed from the posed requirements of a software system. Design is also called *programming on large scale*. The results of the definition phase i.e., the *system definition*, consist of the system requirements specification and the software system model and are the fundament of the design.

Figure 7-3 shows the product flow chart of the design phase.

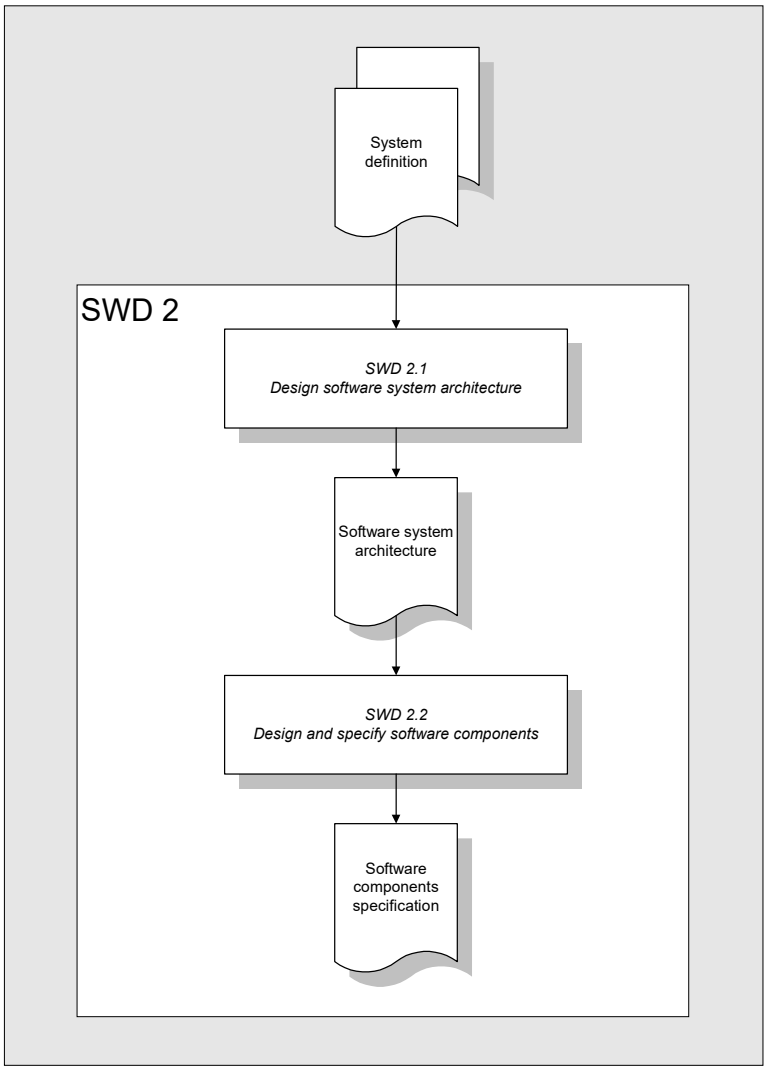

Figure 7-3: Product flow chart of the design phase

7.3.1 Design a Software System Architecture (SWD 2.1)

The software system architecture describes the breakdown of a software system in software components. Suggestions for possible system subdivisions in software components (larger systems are first subdivided into subsystems and/or segments) are described in a rougher form. The individual system functions are assigned to the software components. A short performance description is composed for each software component. The software system architecture is the prerequisite to describe the interfaces between components and modules. This again is the prerequisite for refining, implementing and testing the defined components and modules, if necessary, even by other staff members. If a system architecture consists of many system components, then a more detailed structuring is sensible. Most software applications recommend a three-layer architecture: user interface, actual application, data management. Each layer consists of individual system components. A system component is a contained part of the system. It serves as the unit for the physical structure of a physical application. Examples of software components are functions/procedures, abstract data objects, classes.

The following topics influence decisions on modularization:

- Process structure
- Target computer (run-time system, protocols)
- Concepts of error management, communication, etc.
  Criteria like: information hiding, data abstraction, encapsulation, minimalism of interfaces, manageability, testability, integratable, reusability, etc.
- Requirements on time behavior and reliability

Components of different target directions will arise:

- Control components
- Gateway components
- Data components
- Functional components
- User interface components
- Service components

Note:
An investigation on ready-to-use products can be performed based on the software system architecture. They are tested to compare the necessary modification effort with the "new" development effort. The functionality and interfaces of an off-the-shelf product (components, modules, databases, cards, boards) may require a redesign of the software system architecture.

7.3.2 Software System Architecture

7.3.2.1 Purpose of the Document

The software system architecture describes the structure of the software system with software components and their relations/interfaces between each other. It is represented by a chart. In addition to the list of software components with a short description the document contains a complete definition of the interfaces between the components.

7.3.2.2 Structure Scheme

0 Table of Contents

1 Environmental and Constraint Conditions

2 Basic Design Decisions

3 Diagram of the Software System Architecture

4 Software Components

5 Interface Definitions

5.1 Interface <xxx>*

Amount: ca. 6-10 pages

7.3.3 Design and Specify Software Components (SWD 2.2)

After the software system architecture has been subdivided into components and the interfaces have been identified and defined, the components can be described in detail. These are defined in the specification of the software components. Based on these specifications the individual components can then be implemented independent of each other. The specification can be performed by using a suitable design method like

- Structured Design (SD)
- Modular Design (MD)
- Object-oriented Design (OOD).

7.3.4 Software Components Specification

7.3.4.1 Purpose of the Document

The software components specification defines the functional and performance scale for each component. This document serves as the fundament for the implementation of the components and is very important for maintenance and updates of the system at a later date.

7.3.4.2 Structure Scheme

0 Table of Contents

1 Component <xxx>*

2 Component <xxx>*

Amount: ca. 8-12 pages

7.4 Implementation Phase (SWD 3)

The programming activities are done during the implementation phase. The design is the starting point for the implementation phase. It is assumed that during the design phase a software system architecture was designed, and the respective software components were specified.

Each software component is described in the document *specification of the software components*. Especially the software system architecture should be designed in such a way that the implementations are only a few pages in length.

The implementation phase includes the following activities:

- Conception of data structures and algorithms
- Structuring programs
- Documentation of programs with comments
- Realization in a programming language
- Creation of test programs according to test specification

These activities are also called *programming in detail*.

The results of the implementation phase are all source codes including integrated documentation, all object programs, make files and test programs. Figure 7-4 shows the product flow chart in the implementation phase.

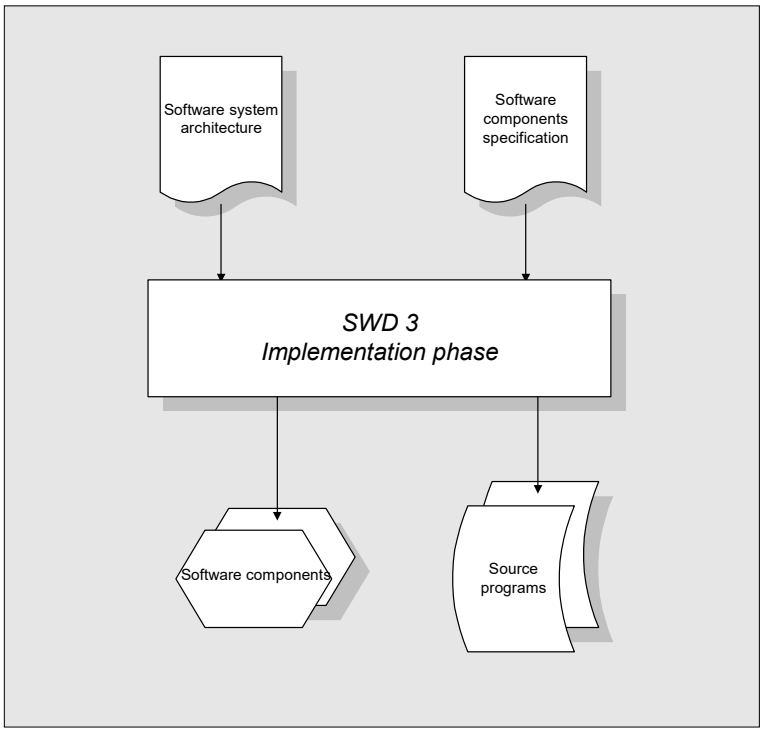

Figure 7-4: Product flow chart in the implementation phase

During implementation several principles should be adhered to. The principle of *integrated documentation* is especially important. Programs should be easy to understand, and the idea of the programmer should be easily recognized. Choosing reasonable identifiers and using suitable name conventions are part of these principles. The integrated documentation must also contain important data like author, version, status, date, etc.

During implementation the principle of *step-by-step refining* is just as important. An algorithm or a program does not "fall from the sky", it needs to be developed systematically. Beginning with a precisely defined problem in the form of a software component specification a solution is formulated with abstract data and abstract instructions. Abstract means problem-related, but not expressed in a concrete programming language notation. This abstract solution is refined incrementally i.e., the abstract instructions are substantiated and step-by-step converted to instructions of the target programming language until the program has been completely written in the target language. The abstract instructions can be used as comments.

7.4.1 Source Programs with Integrated Documentation

The source programs include all completed source files and/or all files needed to generate the software system (source code, make files, configuration files, etc.). A concrete structure is not prescribed for the individual files. But general programming rules should be applied.

7.5 Integration and Acceptance Phase (SWD 4)

Now that the individual software components have been implemented in the implementation phase, they need to be integrated i.e., to be put together. As a rule, integration is done incrementally. After integrating all software components, the complete system is submitted to a system test.

Figure 7-5 shows the product flow chart in the integration and acceptance phase.

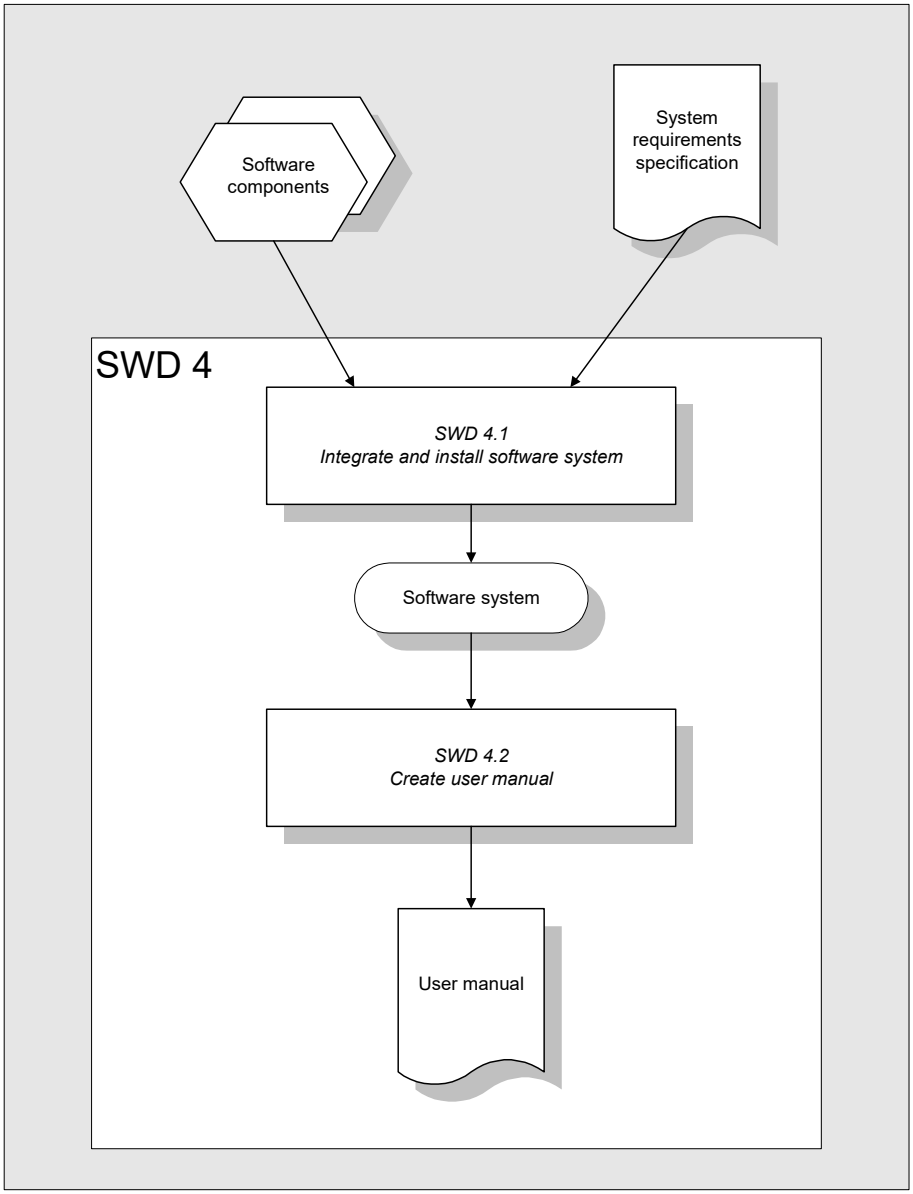

Figure 7-5: Product flow chart in the integration and acceptance phase

7.5.1 Install and Integrate the Software System (SWD 4.1)

All intermediate products of all software components must be integrated and run through a system test. The integrated intermediate products constitute the total system.

7.5.2 Total Installed System

The complete installed, tested and therefore operable system is the end product of the software development process.

7.5.3 Create the User Manual (SWD 4.2)

In order to install and operate the system a manual must be written. This must include the installation steps. Necessary prerequisites for installation and operation are also to be documented. Further the use of several functions should be described in short length. Complex operations should be explained in individual steps.

7.5.4 User Manual

7.5.4.1 Purpose of the Document

The user manual contains all information needed to use the system. This includes installation, operation as well as error correction.

7.5.4.2 Structure Scheme

0 Table of Contents

1 Installation

2 Operation

3 Elements

4 Functions

5 Executing Special Operations

6 Error Correction/Debugging

Amount: ca. 6-10 pages

8 Process Model for System Development Projects

8.1 Objective

This process model class regulates the procedures for advanced undergraduate and thesis projects, that deal with the development of a hardware/software system. For the sake of oversight, the required activities and products are listed in the following.

During system development the following four phases are passed:

- Definition Phase (SD 1)
- Design Phase (SD 2)
- Implementation and Realization Phase (SD 3)
- Integration and Acceptance Phase (SD 4)

Table 8-1 shows an overview of the activities and products in the different phases for Bachelor, Research, Study and Master Theses respectively. Documents marked with an “X” must be completed, those with an “O” are optional and those marked with an “-“ are not required.

Project Phases			BT/RT/ST/MT
SD 1 Definition Phase			
SD 1.1	Determine and define requirements	System requirements specification	X
SD 1.2	Analyze requirements	System model	X
QA 1.2		→ Definition review	X
SD 2 Design Phase			
SD 2.1	Design system architecture	System architecture	X
SD 2.2	Design and specify system components	Specification of system components	X
QA 2.1	Create test specification	Test specification	X
QA 1.3		→ Design review	X
SD 3 Implementation and Realization Phase			
SD 3.1	Create system components	System components Source programs with integrated documentation	X
QA 2.2	Test system components	Test protocol (components)	X
QA 1.4		→ Realization review	X
SD 4 Integration and Acceptance Phase			
SD 4.1	Integrate and install system	Installed system	X
SD 4.2	Create user manual	User manual	X
QA 2.3	Test complete installed system	Test protocol (system)	X
QA 1.5		→ Acceptance review	X

Table 8-1: Overview of activities and products in the model for system development projects

Figure 8-1 shows an overview of the system development. The individual phases are described in detail in the following sections.

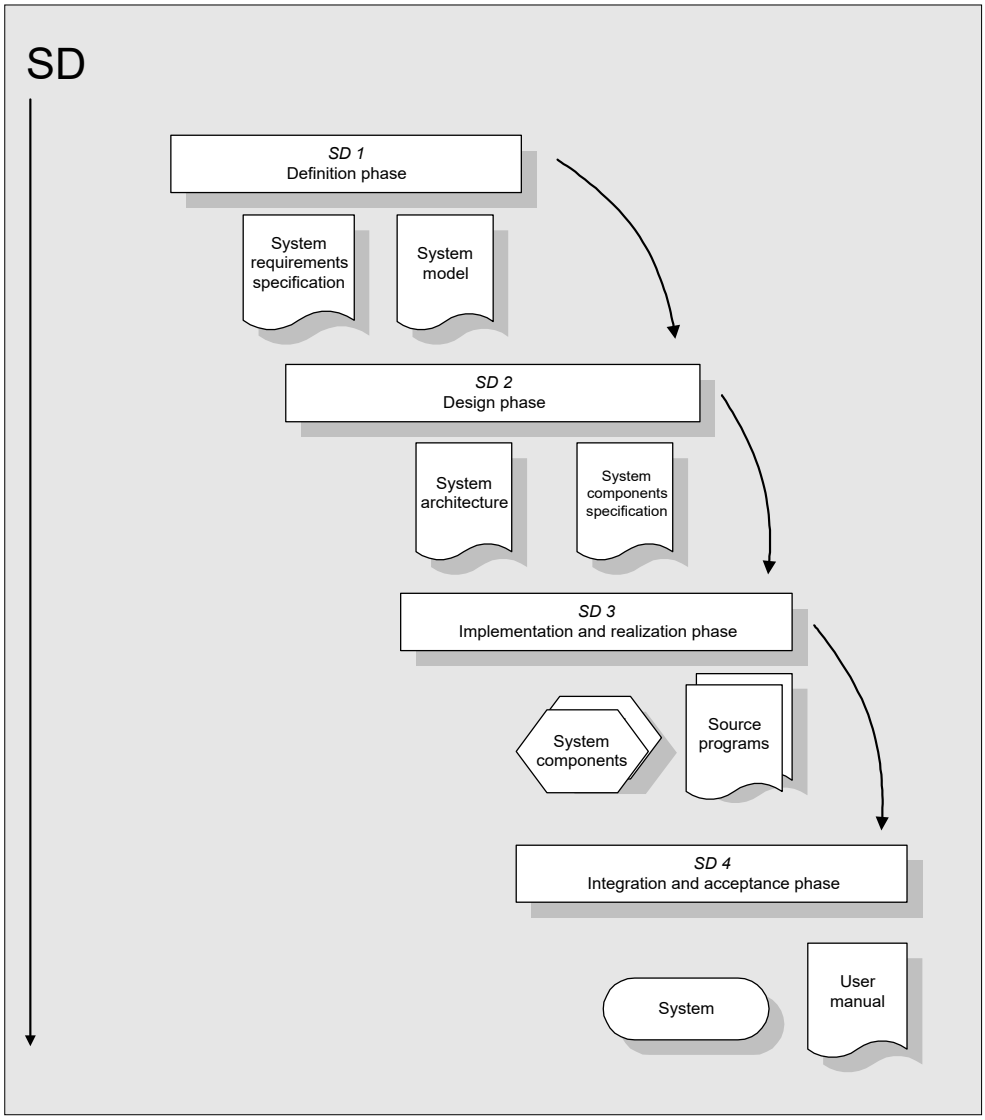

Figure 8-1: Phases and products of the model for system development projects

8.2 Definition Phase (SD 1)

During the definition phase the requirements for a proposed system are defined. The requirements determine the qualitative and quantitative characteristics of a system from the perspective of the contractor. The iterative process of determining the requirements is called system analysis. The *system definition* is the fundament for the acceptance of the system and is therefore very important. The system definition consists of the products *system requirements specification* and *system model*.

Figure 8-2 shows the product flow chart in the definition phase.

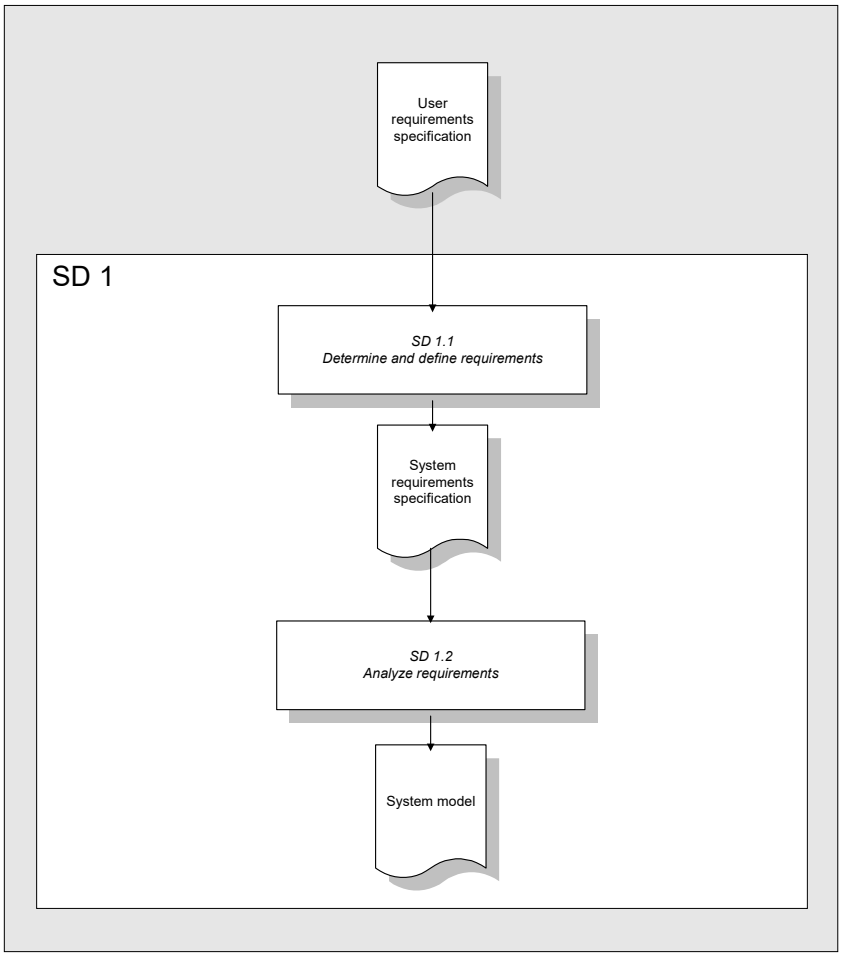

Figure 8-2: Product flow chart of the definition phase

8.2.1 Determine and Define Requirements (SD 1.1)

The basic requirements for the proposed system described in the user requirements specification must be determined in more detail. Normally this is done by surveys and discussions with the supervisor and possible users. If an existing system is to be improved or replaced, then a current status analysis of the present system should be completed first. This can often reveal weaknesses and flaws which can be taken into consideration when drafting the new requirements.

After the requirements have been determined they need to be described and defined. For this purpose, they are classified and structured into several categories. Functional and non-functional requirements are differentiated. Depending on the type and complexity of the system a further subdivision may be necessary. In addition to functional and non-functional requirements further requirements regarding quality must be defined. Also, development and target environments need to be determined. All requirements must be written down in the system requirements specification.

8.2.2 System Requirements Specification

8.2.2.1 Purpose of the Document

The system requirements specification is a summary of all academic requirements, that the proposed system needs to fulfill from the perspective of the contractor. It contains the academic scope of the system in reference to its function, data, performance and quality. The system requirements specification defines *what* the system is to perform and not *how*.

8.2.2.2 Structure Scheme

0 Table of Contents

1 Objective

- 1.1 Mandatory Criteria
- 1.2 Optional Criteria
- 1.3 Limit Criteria

2 Operation

- 2.1 Areas of Application
- 2.2 User Group
- 2.3 Operating Conditions

3 Environment

- 3.1 Software
- 3.2 Hardware
- 3.3 System Interfaces

4 Functional Requirements

5 Non-functional Requirements

6 Requirements for the user interface

7 Quality Requirements

8 Global Test Scenarios and Test Cases

9 Development Environment

- 9.1 Software
- 9.2 Hardware
- 9.3 Development Interfaces

10 Execution

11 Additions

Amount: ca. 8-15 pages

8.2.3 Analyze Requirements (SD 1.2)

Analyzing the requirements is often not enough to attain a sufficiently clear concept of the system to be developed. A system model forces an academic approach to the problem which is not possible if the requirements are stated verbally.

Analysis occurs according to a suitable analysis method. We recommend using the analysis method *Use Cases*. In agreement with the supervisor another method may be used.

The desired functionality of the hardware part should be analyzed with the help of block diagrams. The requirements on the function blocks are to be described in a suitable manner.

8.2.4 System Model

8.2.4.1 Purpose of the Document

Creating a verbal system requirements specification is not sufficient, therefore a semi-formal or formal system model must be created. The system model describes the academic concept completely, unambiguously and consistently. The contractor (i.e., the supervisor) should be able to understand the system model, he does not need to create it himself.

8.2.4.2 Structure Scheme

0 Table of Contents

1 Software System Model

2.1 Use Case Overview
2.2 Use Case Diagram
2.3 Description of Use Case <xxx>*
2.4 Description of Use Case <xxx>*
2.5 Sequence Diagram of Use Case <xxx>*

2 Hardware System Model

2.1 Block Diagram
2.2 Description of functional blocks
2.2.1 Functional block <xxx>*
2.2.2 Functional block <xxx>*

Amount: ca. 6-10 pages

8.3 Design Phase (SD 2)

In the design phase a systematic solution in the form of a *system architecture* is developed from the requirements on the system. The results of the definition phase i.e., the *system definition* (system requirements specification and system model), are the fundament for the design. Figure 8-3 shows the product flow chart in the design phase of system development.

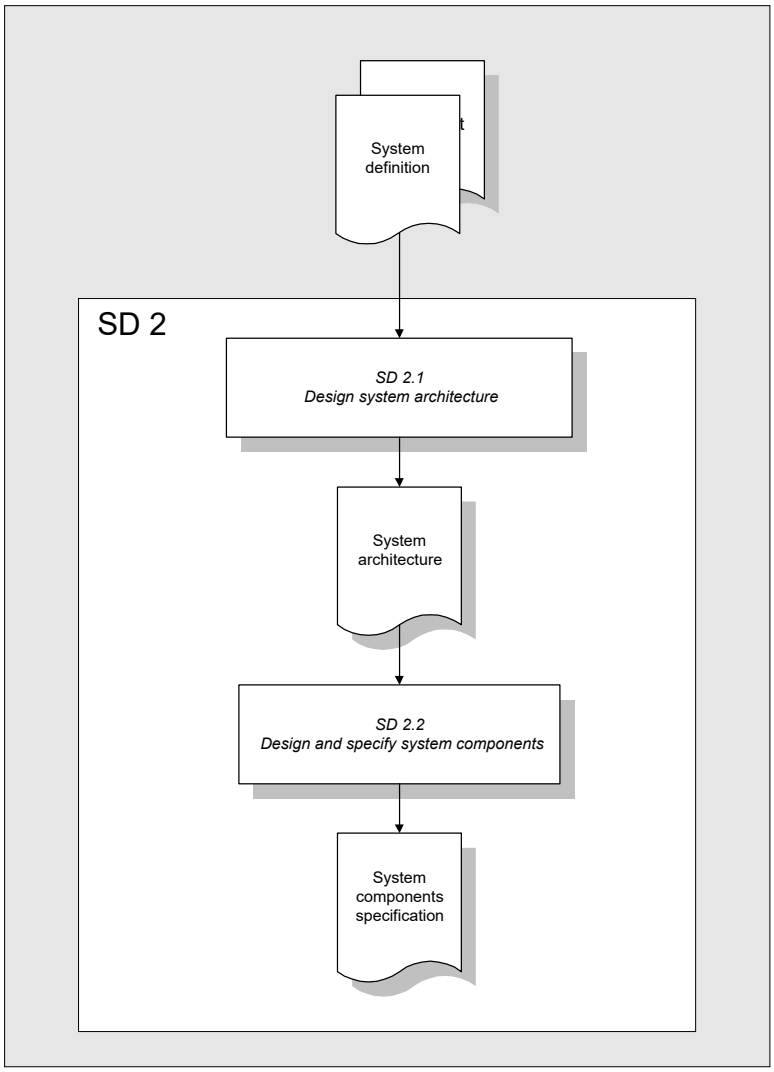

Figure 8-3: Product flow chart in the design phase

8.3.1 Design a System Architecture (SD 2.1)

First the constraint and environment conditions for the proposed system are listed and theoretical solution approaches are discussed. In this phase fundamental design decisions are made on *how* the system is to be realized.

8.3.2 System Architecture

8.3.2.1 Purpose of the Document

The system architecture describes the structure of the system with its system components and their interfaces to each other. This document describes both the software and hardware system architectures. In addition to a list of components with a short description the document also contains a complete definition of the interfaces between the components especially between the hardware and the software system.

The system architecture describes the breakdown of a system in its system components. Suggestions are described for possible breakdowns of the system into components (larger systems are first divided into subsystems and/or segments). The individual system functions are assigned to the system components. Each component is to be described in a short performance

description. The system architecture is the prerequisite for describing the interfaces between components and modules. This is the prerequisite to allow for a systematic refining, implementation, realization and testing of components and modules, possibly by various project co-workers.

Software System Architecture
If a system architecture consists of many system components, a detailed structure is reasonable. For most software applications a three-layer architecture is recommended: user interface, actual application, data management. Each layer in turn consists of individual system components. A system component is a contained part of the system. It serves as a unit for the physical structure of an application. Examples for software components are functions/procedures, abstract data objects, classes.

The following influence modularization decisions e.g.:

- Process structure
- Target computer (run-time system, protocols)
- Concepts of error management, communication, etc.
- Criteria like: information hiding, data abstraction, encapsulation, minimalism of interfaces, manageability, testability, integratability, reusability, etc.
- Requirements on time behavior and reliability

Components of different target directions will arise:

- Control components
- Gateway components
- Data components
- Functional components
- User interface components
- Service components

Hardware System Architecture
The hardware system architecture is described with block diagrams (diagram of the system architecture). In addition to a list of components with a short description the document also contains a complete definition of the interfaces between the components i.e., the wiring schematics, plugs and connectors, etc. are listed here.

8.3.2.2 Structure Scheme

Amount: ca. 8-12 pages

8.3.3 Design and Specify System Components (SD 2.2)

After the system has been broken down into components in the system architecture and interfaces have been identified and defined, the components must be described in detail. They are defined in the specification of the system components. Based on these specifications the individual components can be realized independently of each other.

The specification of the software components can be done by employing suitable design methods like

- Structured Design (SD)
- Modular Design (MD)
- Object-oriented Design (OOD).

Specification of the hardware components is done by drawing wiring schematics, dimensioning components, designing layouts, completing component lists etc.

8.3.4 System Components Specification

8.3.4.1 Purpose of the Document

The specification of the system components determines the function and performance capabilities for each component. This document serves as a basis for the realization of the components and is especially important for maintenance and repair of the system later during operation. This document specifies both hardware and software components.

8.3.4.2 Structure Scheme

0 Table of Contents

1 Hardware Component <xxx>*

1.1 Circuit diagram <xxx>*

1.2 Component sizing<xxx>*

1.3 Layout <xxx>*

1.4 Component list <xxx>*

2 Hardware Component <xxx>*

...

3 Software Component <xxx>*

4 Software Component <xxx>*

Amount: ca. 8-12 pages

8.4 Realization Phase (SD 3)

In the realization phase the hardware system and the software system are implemented. The design is the fundament for this phase. It is assumed that a system architecture and the corresponding requirements were completed in the design phase.

The hardware realization phase includes the following activities:

- Completing layout transparencies
- Manufacturing printed circuit boards (PCBs)
- Equipping the PCBs
- Wiring the connections
- Installment in the cabinet/housing
- Creating a front plate
- Testing the completed hardware components according to test specification

The programming activities are done during the implementation phase. The implementation phase includes the following activities:

- Conception of data structures and algorithms
- Structuring programs
- Documentation of programs with comments
- Realization in a programming language
- Creation of test programs according to test specification

These activities are also called *programming in detail*.

The results of the realization phase are the hardware components, all source programs including integrated documentation, all object programs, make files and test programs.

8.4.1 Source Programs with integrated Documentation

The source programs include all completed source files and/or all files needed to generate the software system (source code, make files, configuration files, etc.). A concrete structure is not prescribed for the individual files. But general programming rules should be applied.

Figure 8-4 shows the product flow chart in the realization phase.

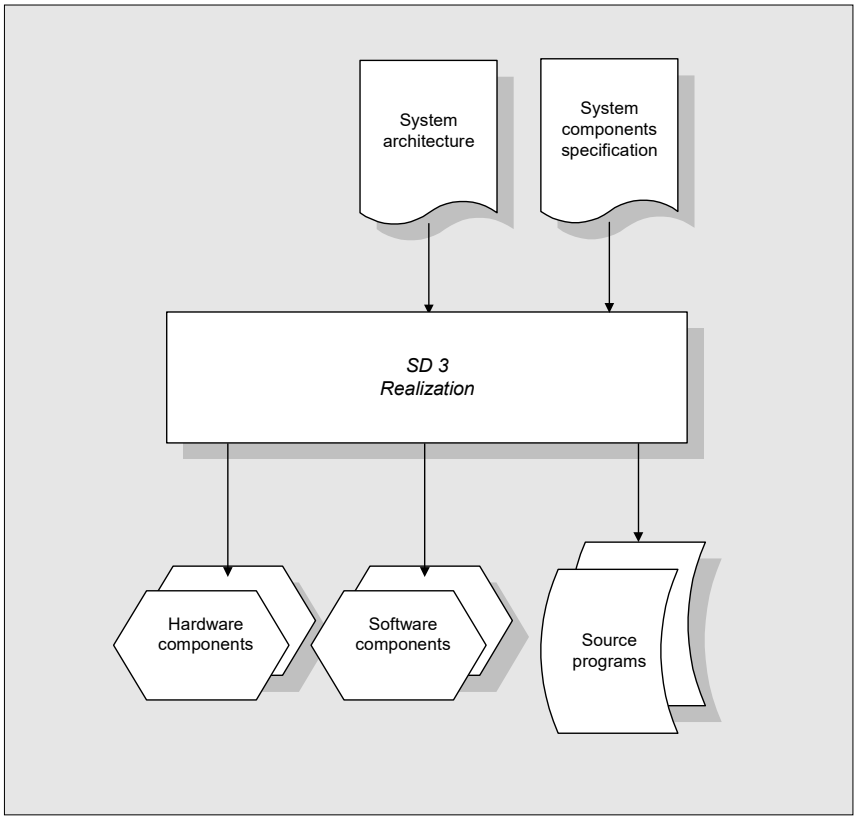

Figure 8-4: Product flow chart in the realization phase

8.5 Integration and Acceptance Phase (SD 4)

If the individual system components have been manufactured or implemented in the realization phase, they need to be integrated i.e., assembled. Integration is normally done step-by-step.

After all components have been integrated the complete system is subjected to a system test (cp. section 8.5.1).

Figure 8-5 shows the product flow chart in the integration phase.

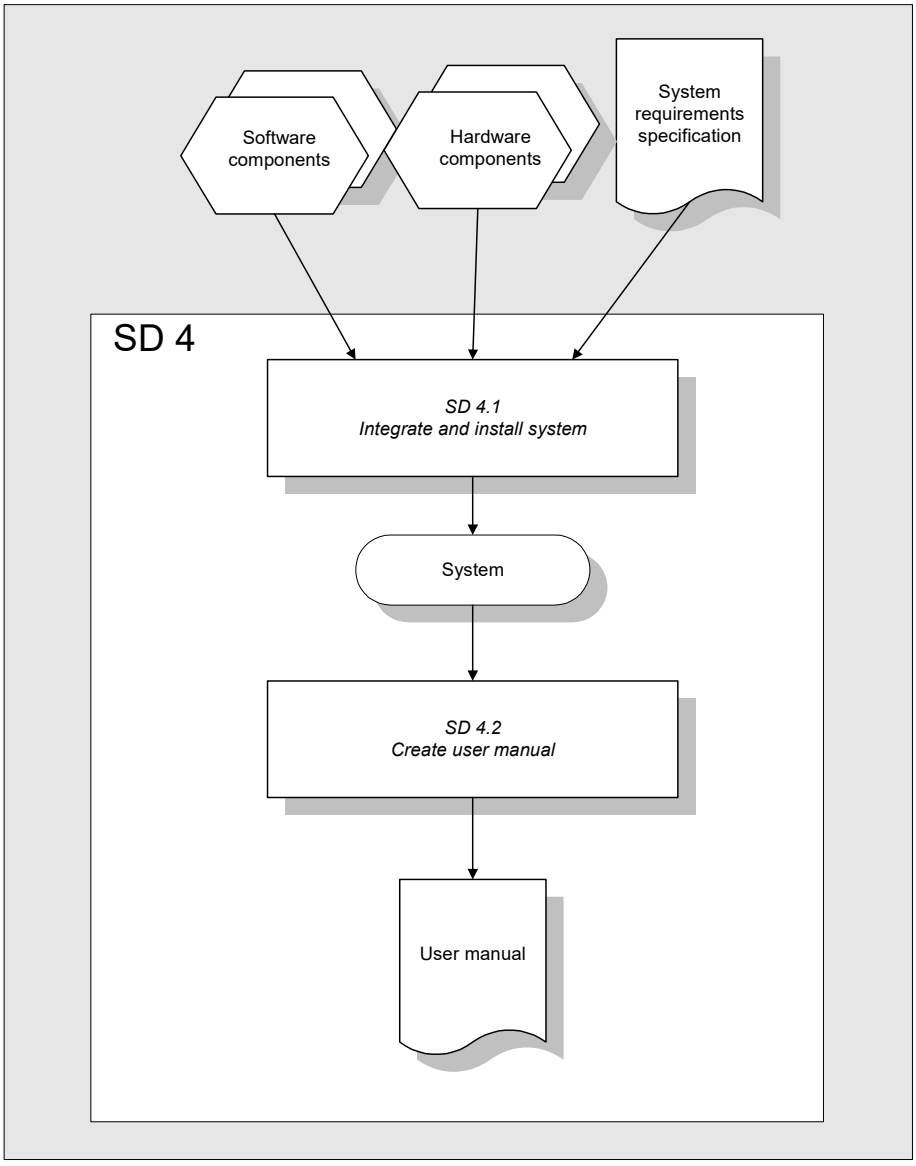

Figure 8-5: Product flow chart in the integration and acceptance phase

8.5.1 Integrate and Install the System (SD 4.1)

All intermediate products of all system components must be integrated and subjected to a system test. The integrated intermediate products constitute the total system.

8.5.2 Total Installed System

The complete installed, tested and therefore operable system is the end product of the system development process.

8.5.3 Create the User Manual (SD 4.2)

In order to install and operate the system a manual must be written. This must include the installation steps. Necessary prerequisites for installation and operation are also to be

documented. Further the use of several functions should be described in short length. Complex operations should be explained in individual steps.

8.5.4 User Manual

8.5.4.1 Purpose of the Document

The user manual contains all information needed to use the system. This includes installation, operation as well as error correction.

8.5.4.2 Structure Scheme

0 Table of Contents

1 Installation

2 Start-up

3 Overview over the Control Elements

4 Functions

5 Execution of special Operations

6 Error correction

Amount: ca. 6-10 pages

9 Model for Conceptional Projects

9.1 Objective

This process model class defines the process of advanced undergraduate and thesis projects, which deal with the completion of theoretical concepts. These may be concepts for software or software/hardware systems. This model also provides for the creation of a prototype (optional). A prototype does not necessarily need to fulfill the high-quality standards, as demanded for completed systems in the first two process model classes. The prototype realization can be used to complete and/or evaluate the concept.

The concept development consists of the following three phases:

- Definition Phase (CD 1)
- Conception Phase (CD 2)
- Prototyping Phase (CD 3)

Table 9-1 shows an overview of the activities and products in the different phases for Bachelor, Research, Study and Master Theses respectively. Documents marked with an “X” must be completed, those with an “O” are optional and those marked with an “-“ are not required.

Project Phases	**BT/RT/ST/MT**

CD 1 Definition Phase			
CD 1.1	Analyze and define requirements for the concept	System requirements specification	X
CD 1.2	Develop the basics	Basis	X
QA 1.2		→ Definitions review	X
CD 2 Conception Phase			
CD 2.1	Design the concept	Concept	X
QA 2.1	Create evaluation specifications	Evaluation specifications	O
QA1.3		→ Design review	X
CD 3 Prototyping Phase			
CD 3.1	Create a prototype	Description of the prototype	O
QA 2.2	Perform evaluation	Evaluation protocol	O
QS1.4		→ Prototyping review	O
QS1.5		→ Acceptance review	X

Table 9-1: Overview of the activities and products in the model for conceptional projects

Figure 9-1 shows an overview of the concept development. The individual phases are described in detail in the following chapters. The prototyping phase can be done during or after the definition and/or conceptional phases.

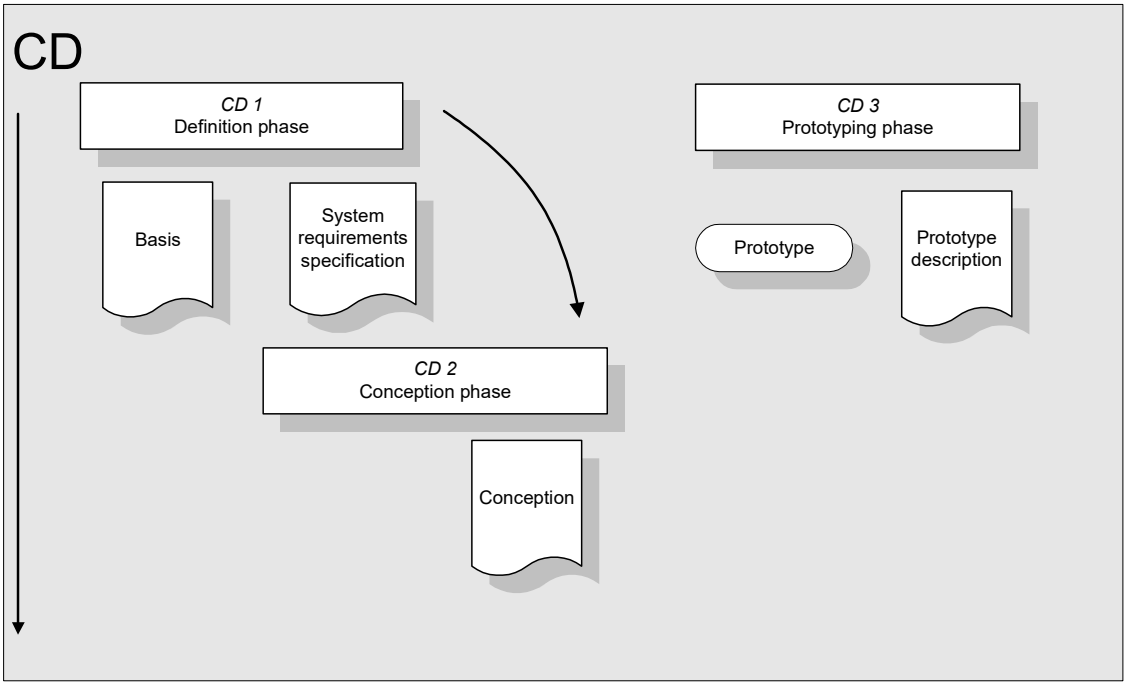

Figure 9-1: Phases and products of the model for conceptional projects

9.2 Definition Phase (CD 1)

In the definition phase, the requirements for the proposed concept are defined. The requirements determine the qualitative and quantitative characteristics from the perspective of the contractor. This definition is the basis for the acceptance of the concept and is therefore very important. The definition consists of the product *system requirements specification*. The definition process is iterative. Normally several repetitions of the individual activities are necessary. The results of the definition phase are complete, unambiguous, consistent and feasible requirements for the proposed concept.

Figure 9-2 shows the product flow chart of the definition phase.

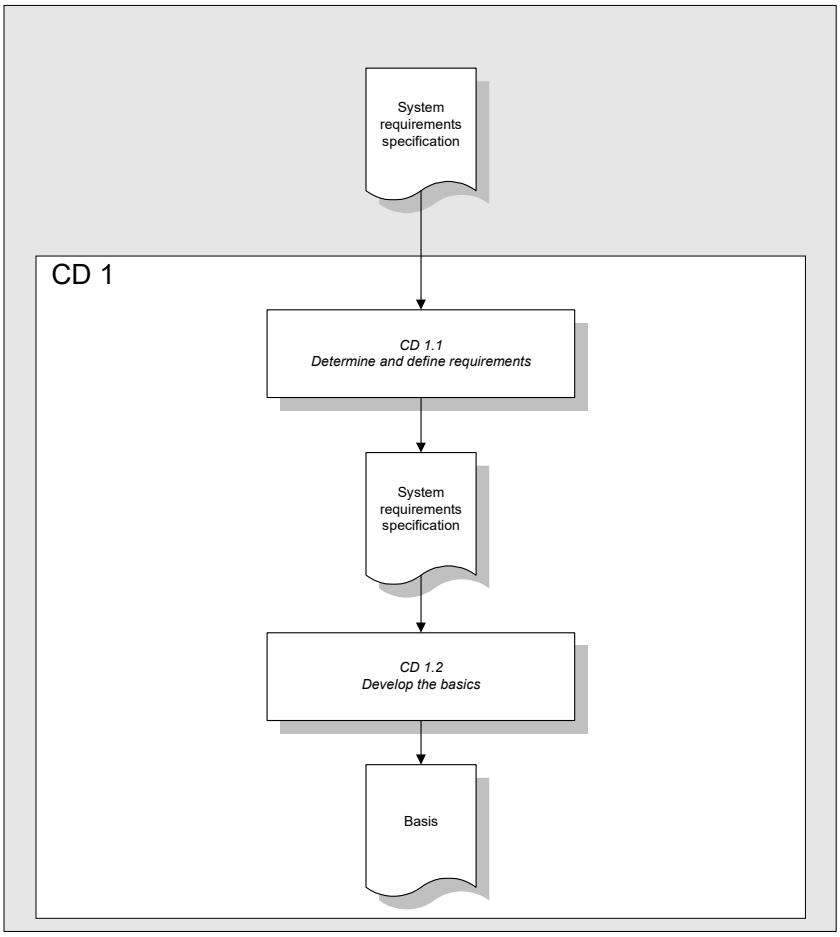

Figure 9-2: Product flow chart in the definition phase

9.2.1 Analyze and Define Requirements (CD 1.1)

The basic requirements of the proposed concept described in the user requirements specification must be determined in more detail. Normally this is done by surveys and discussions with the supervisor and the contractor. After the requirements have been determined, they are described and defined. For this purpose, they are classified and structured into several categories. Depending on the type and complexity of the subject matter a further subdivision may be necessary. It can be very helpful to develop the concept using a prototypical realization or to evaluate it using a prototype. In this case the requirements on the prototype must be specified in the system requirements specification, the development and target environments must be determined as well.

9.2.2 System Requirements Specification

9.2.2.1 Purpose of the Document

The system requirements specification is a summary of all academic requirements, that the proposed concept needs to fulfill from the perspective of the contractor. It contains the academic scope of the concept in reference to its performance and quality. The system requirements specification determines what the proposed concept must contain.

9.2.2.2 Structure Scheme

0 Table of Contents

1 Objective

1.1 Mandatory Criteria
1.2 Optional Criteria
1.3 Limit Criteria

2 Operational Area

2.1 Areas of Application
2.2 User Group

3 Requirements on the Concept

4 Quality Requirements

5 Execution

6 Prototype*

6.1 Environment
6.1.1 Software
6.1.2 Hardware
6.1.3 System Interfaces
6.2 Quality Requirements
6.3 Development Environment
6.3.1 Software
6.3.2 Hardware

7 Global Evaluation Methods

8 Additions

Amount: ca. 8-15 pages

9.2.3 Develop the Basics (CD 1.2)

Prerequisite for creating the concept is the development of the necessary fundamentals. Depending on the subject matter of the project this can be done by processing literature, specifications, publications, etc. or by establishing formulas and equations. The objective of this phase is to analyze, compare and evaluate various concept ideas. Furthermore, basic technologies which are important for the thesis are explained. The phase ends with a decision for one certain approach.

9.2.4 Basis

9.2.4.1 Purpose of the Document

In this document the fundamentals for the proposed concept are listed. Therefore, it is the fundament for the proposed concept. It may also contain a comparison of different approaches. For this case the following structure scheme is presented.

9.2.4.2 Structure Scheme

0 Table of Contents

1 Environment and Constraint Conditions

2 Approach <xxx>*

2.1 Description
2.2 Advantages
2.3 Disadvantages
2.4 Summary

3 Approach <xxx>*

3.1 ...

4 Comparison of Approaches

5 Basic Decisions

6. Technological Basics*

6.1 <Technology 1>

Amount: ca. 10-20 pages

9.3 Conception Phase (CD 2)

The concept is created in the conception phase. The fundamentals for this phase are the requirements in the system requirements specification and the basics determined during the definition phase. Figure 9-3 shows the product flow chart in the conception phase.

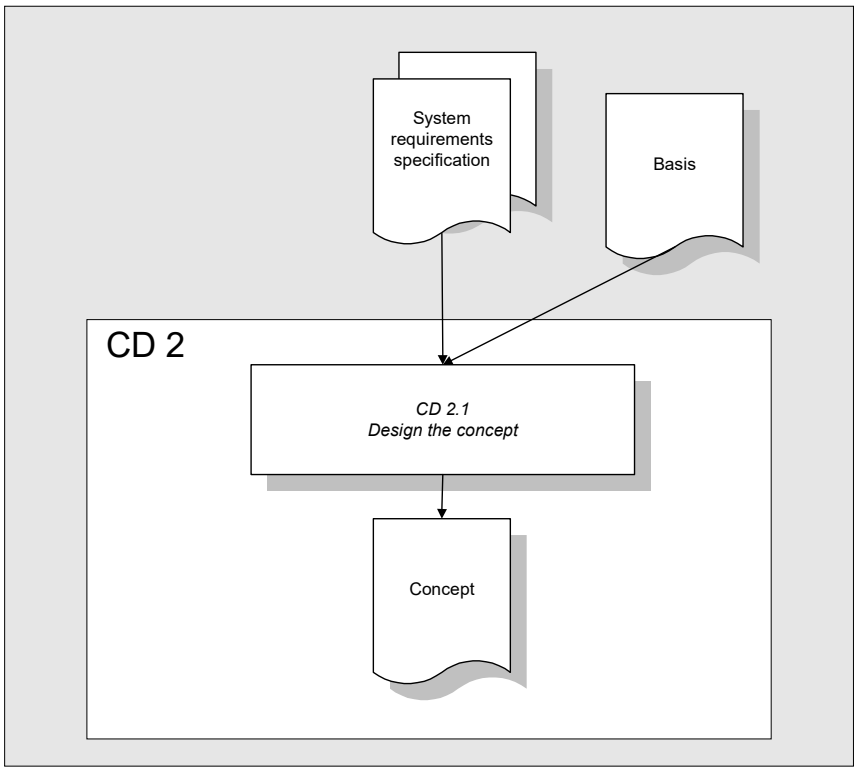

Figure 9-3: Product flow chart in the conception phase

9.3.1 Design the Concept (CD 2.1)

This project phase contains the actual concept design, after all the basics have been developed for this step. The process model does not prescribe a certain procedure for this phase, it is the responsibility of the student and the supervisor to select a suitable method. The structure scheme is therefore to be designed independently.

9.3.2 Concept

9.3.2.1 Purpose of the Document

This document presents the developed concept. A clear presentation is important.

9.3.2.2 Structure Scheme

0 Table of Contents

1 Description of the Concept

1.1 Sub-section *
1.2 ...

2 Summary

Amount: ca. 10-20 pages

9.4 Prototyping Phase (CD 3)

The prototyping phase is optional. In the chapter *execution* of the user requirements specification determines if a prototyping phase is to be included in the project.

9.4.1 Create a Prototype (CD 3.1)

Depending on the type of work being done, a prototype can be helpful. A prototype can be used to analyze the requirements of the concept or to develop the concept with the help of the prototype. For such cases this phase is executed parallel to the definition phase or the conception phase.

Alternatively, a prototype can be used to evaluate the completed concept. In this case the prototyping phase is completed after the conception phase, where the concept itself is created. The concept is evaluated using the prototype after the conception phase.

9.4.2 Description of the Prototype

9.4.2.1 Purpose of the Document

This document contains a short description of the completed prototype, that lists all important characteristics of the prototype. The structure scheme is adjusted towards the products, which are created in the model for system development. The individual sections do not need to be processed in such detail as is expected in the model for system development.

9.4.2.2 Structure Scheme

0 Table of Contents

1 System Architecture

2 Description of System Components

3 User Manual

Amount: ca. 8 - 15 pages

9.5 Quality Assurance (QA)

The submodel quality assurance was presented in part I on page 29. The following describes changes to quality assurance necessary for the model for conceptional projects.

9.5.1 Process Testing (QA 1)

9.5.1.1 Definition Review (QS 1.2)

In addition to the review of the system requirements specification, the basis is checked. Figure 9-4 shows the product flow chart of the definition review.

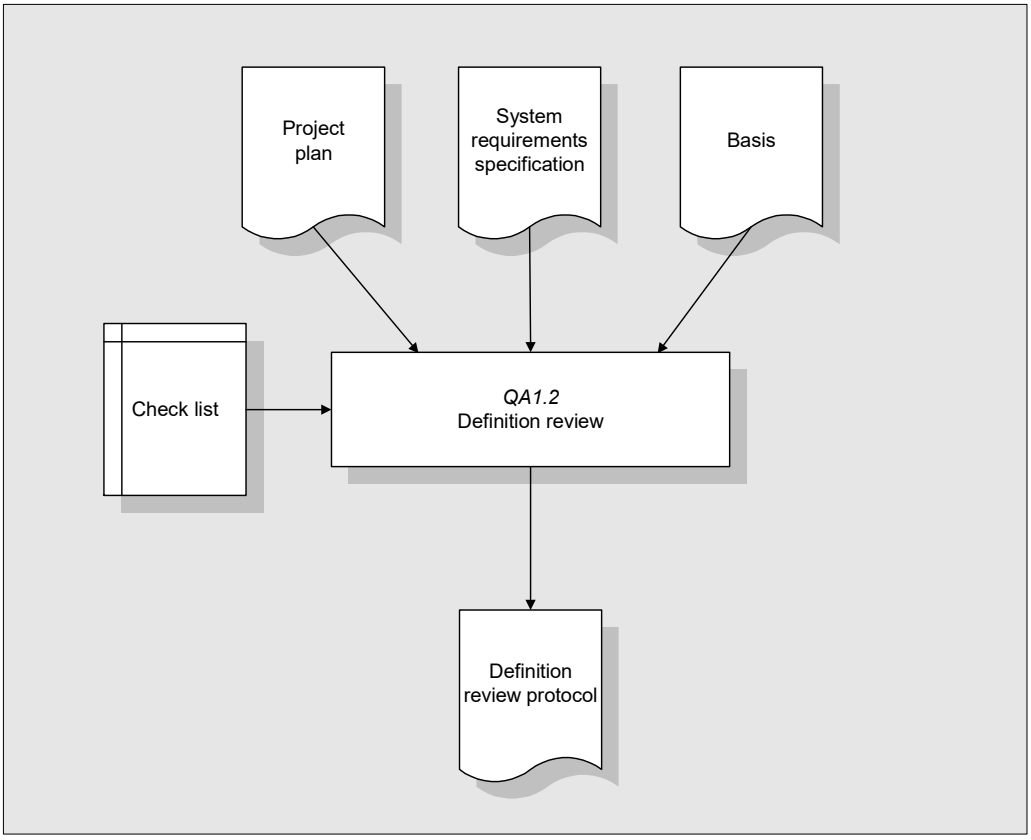

Figure 9-4: Product flow chart of the definition review

9.5.1.2 Design Review (QA 1.3)

The design review tests the *concept* and the *evaluation specification*. Additionally, some re-testing derived from the definitions review may be executed. Figure 9-5 shows the product flow chart of the design review.

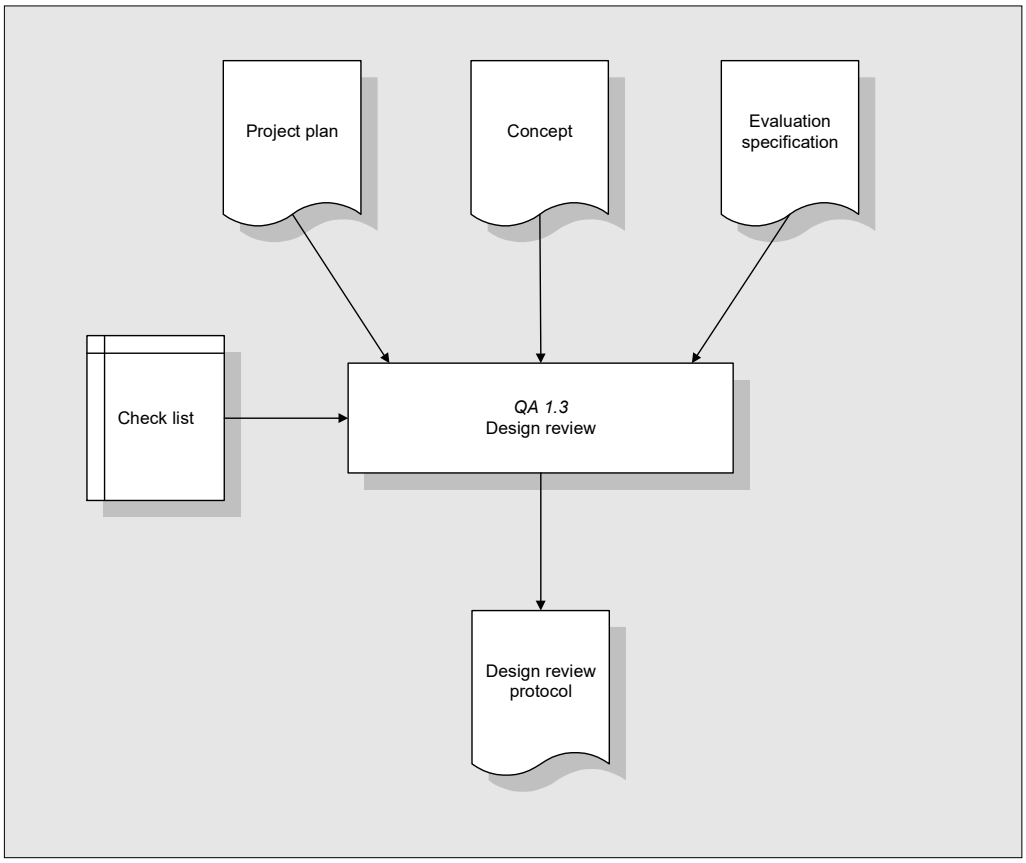

Figure 9-5: Product flow chart of the design review

9.5.1.3 Prototyping Review (QA 1.4)

In the prototyping review the completed *prototype* and the *description of the prototype* are tested. Figure 9-6 shows the product flow chart for the prototyping review.

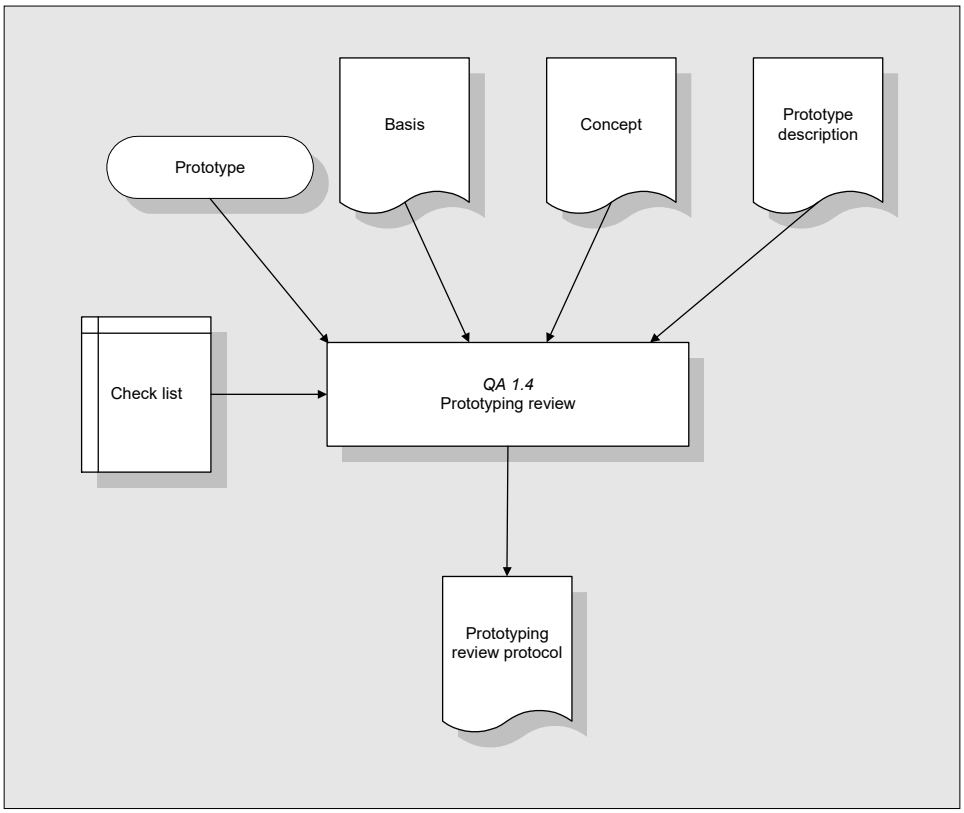

Figure 9-6: Product flow chart for the prototyping review

9.5.1.4 Acceptance Review (QA 1.5)

When the concept is to be accepted the results are checked by the supervisor and the contractor. The results of this review are recorded in the *acceptance review protocol*. Figure 9-7 shows the product flow chart of the acceptance review.

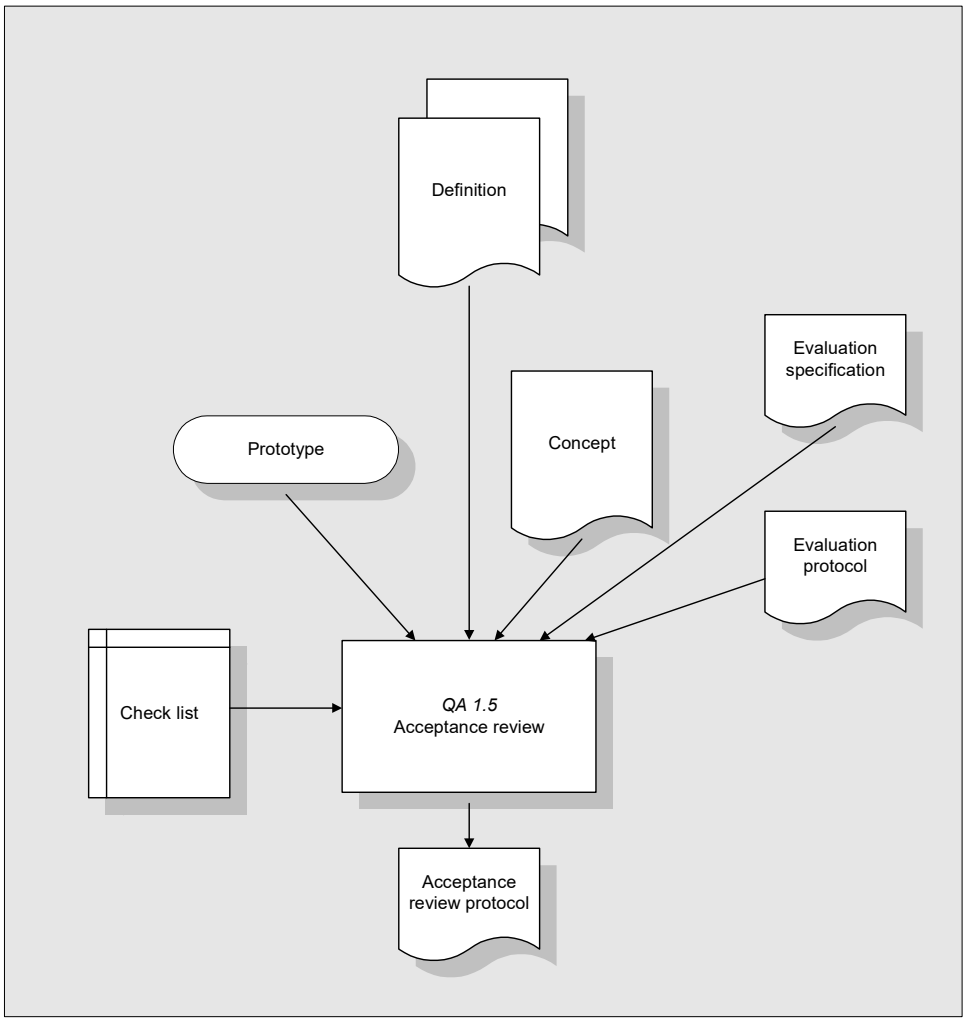

Figure 9-7: Product flow chart of the acceptance review

9.6 Evaluate the Concept (QA 2)

Evaluation of the concept is optional and is determined by the supervisor in the user requirements specification. The evaluation of a concept serves to prove that the completed concept fulfills the specified requirements and the expectancies of the contractor. For this purpose, the concept is evaluated and judged using examples. If a prototype was created, it is used for the purpose of evaluation. Figure 9-8 shows an overview of the evaluation.

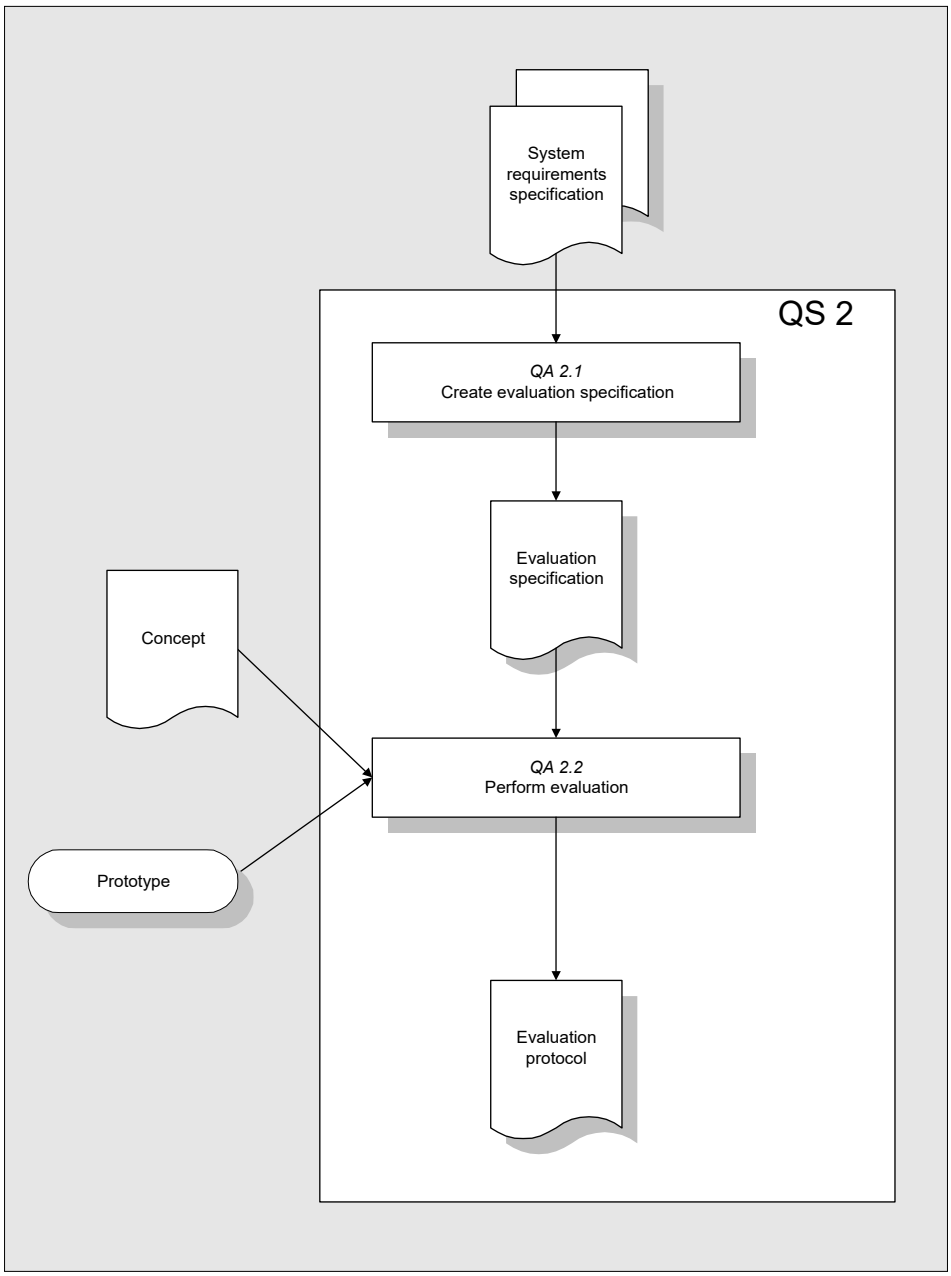

Figure 9-8: Overview of the concept evaluation (QA 2)

9.6.1 Create Evaluation Specifications (QA 2.1)

General requirements for evaluation are derived from the requirements for the proposed concept. After that methods and criteria are determined for evaluation, which fulfill these requirements.

In the framework of advanced undergraduate and thesis projects evaluation cases do not need to be complete for time reasons. The global evaluation methods mentioned in the system requirements specification must be used and specified in detail.

The evaluation specification should be completed before the conception starts.

9.6.2 Evaluation Specification

9.6.2.1 Purpose of the Document

This document contains a description of the requirements, the evaluation goals, applied test methods, requirement-derived evaluation criteria and the evaluation cases. With the help of the evaluation specification a decision can be made on whether the evaluation was successful.

9.6.2.2 Structure Scheme

0 Table of Contents

1 Evaluation Requirements

2 Evaluation Methods

3 Evaluation Criteria

4 Evaluation Cases

Amount: ca. 4-8 pages

9.6.3 Perform Evaluation (QA 2.2)

According to the evaluation cases defined in the evaluation specification the individual sub-sections of the concept are tested by the student. For this purpose, environments must be created for evaluation and applied procedures. The results and evaluations of the individual tests are to be documented in the evaluation protocol.

An orderly test execution is to be supervised by the supervisor with the help of spot tests.

9.6.4 Evaluation Protocol

9.6.4.1 Purpose of the Document

The evaluation protocol contains the records made by the tester on the sequence of the individual tests, especially the comparison of the expected and the actual results.

9.6.4.2 Structure Scheme

0 Table of Contents

1 <Evaluation Case 1>*

1.1 Reference to Evaluation Specification
1.2 Procedure of Testing
1.3 Results
1.4 Discussion

Amount: ca. 4-8 pages

10 Open Model

10.1 Objective

This process model class defines the process of advanced undergraduate and thesis projects, that cannot be classified in one of the other classes and are only rarely executed. For these reasons an extensive regulation is not reasonable. Here we will try to draw a basic framework for such projects to regulate their process. Within this framework the supervisor has the responsibility of defining the procedure in detail in order to gain a high quality of work.

10.2 Submodel Project Management (PM)

The submodel project management determines the exact process of the student's project i.e., activities are defined, which have intermediate results called products. In the section *execution* of the user requirements specification this process is defined.

As in the other process model classes that are described in the first part of this process manual organizational planning, effort and schedule planning as well as resource planning need to be executed (see page 26).

10.3 Submodel Project Execution (PE)

Figure 10-1 shows the project phases of the open model. Generally, four project phases are passed that have at least one intermediate product as a result and are closed with a review. The fundament of each project is the *user requirements specification*, which summarizes all basic academic requirements. After that the definition phase takes place in which the requirements of the project are analyzed and defined. The result of this definition phase is the product *system requirements specification.* The actual processing of the task is done in the following phases. The activities defined in the user requirements specification are to be executed and products created.

10.4 Submodel Quality Assurance (QA)

Quality Assurance consists of two parts, the process testing in which spot tests are performed to ensure that activities adhere to agreed rules and the test/evaluation which proves if the completed project adheres to the specified requirements and the expectancy of the contractor.

10.4.1 Process Testing (QA 1)

Suitable *reviews* are to be planned and performed. Objective of a review is to identify errors as well as to recognize violations against specifications, standards and plans. The results of the individual tests are recorded in the review protocol. No checklists for (intermediate) products are provided for this process model.

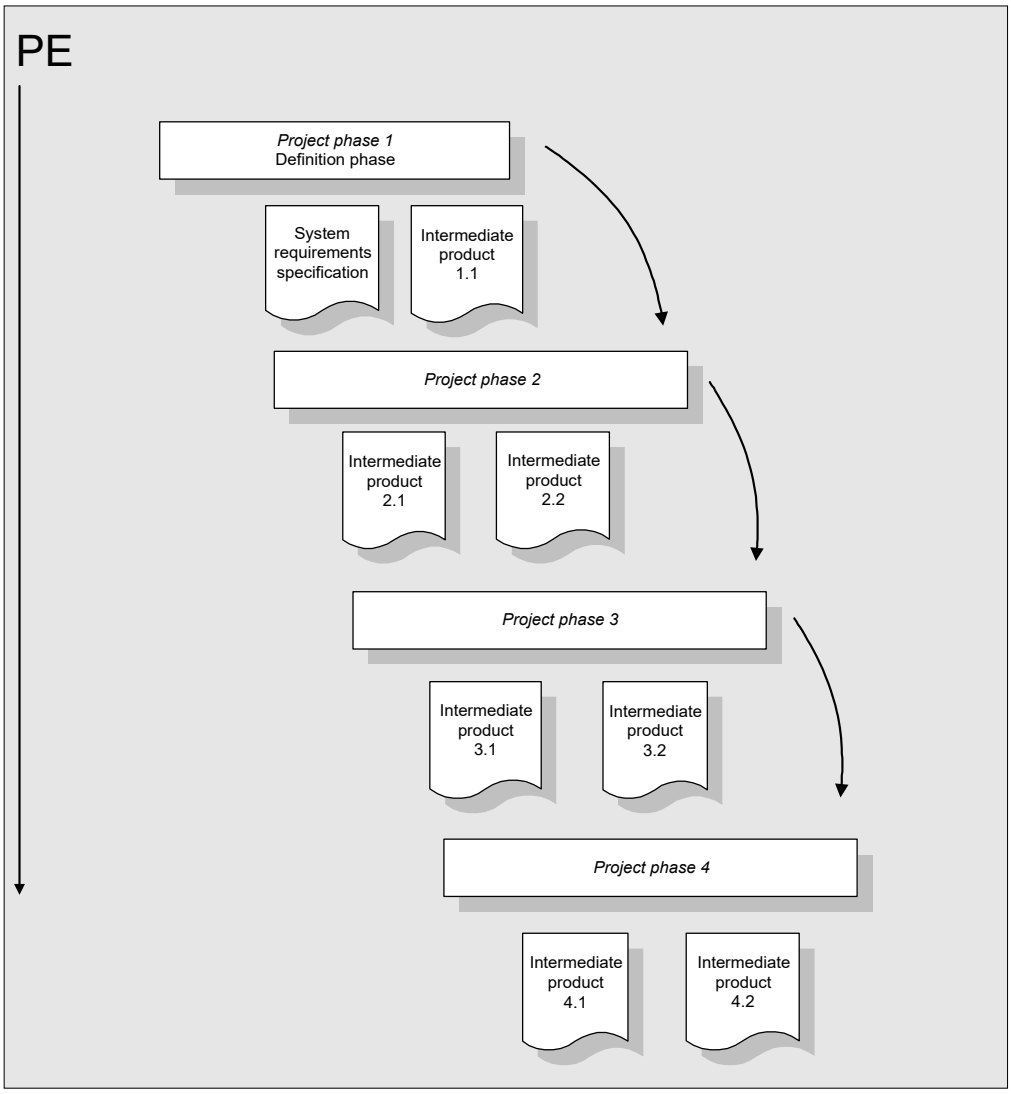

Figure 10-1: Project phases of the open model

10.4.2 Test / Evaluation (QA 2)

The test and evaluation serve to prove that the completed project fulfills the specified requirements and the expectancy of the contractor. Suitable test cases are to be defined and applied to the test object to prove that the test object contains no errors for the defined test cases.

10.5 Submodel Configuration Management (CM)

Configuration management is to be performed as described on page 23. The configuration management archives and indexes all objects of a configuration (documents, protocols, source code files etc.), so that no object can be accidentally or arbitrarily destroyed. All objects are unambiguously identifiable. Therefore, changes made in the development process or during operation can be traced and a defined point for further version changes can be identified.

Table 10-1 provides an overview of the activities and products in the different phases for Bachelor, Research, Study and Master Theses respectively. Documents marked with an “X” must be completed, those with an “O” are optional and those marked with an “-“ are not required.

Project Phases			BT/RT/ST/MT
PE 1 Definition Phase			
PE 1.1	Determine and define requirements	System requirements specification	X
PE 1.1	Intermediate product 1.1		O
QA 1.2		→ Definitions review	X
PE 2 Project Phase 2			
PE 2.1	Intermediate product 2.1		X
PE 2.2	Intermediate product 2.2		O
QA 1.3		→ Review 2	X
PE 3 Project Phase 3			
PE 3.1	Intermediate product 3.1		X
PE 3.1	Intermediate product 3.2		O
QA 1.4		→ Review 3	X
PE 4 Project Phase 4			
PE 4.1	Intermediate product 4.1		X
PE 4.2	Intermediate product 4.2		O
QS1.4		→ Acceptance review	X

Table 10-1: Overview of activities and products of the open model

10.6 Example: Market Overview

10.6.1 Objective

The goal of the market overview is to develop a recommendation or decision aid for the contractor. Such a study could include a comparison of tools present in the market.

10.6.2 Project Phases

The following table shows the project phases and the corresponding activities.

Project Phases			BT/RT/ST/MT
PE 1	**Definition Phase**		
PE 1.1	Determine and define requirements of the study	System requirements specification	X
PE 1.2	Set up a criteria catalog	Criteria catalog = Intermediate product 1.1	X
QA 1.2		→ Definitions review	X
PE 2	**Research Phase**		
PE 2.1	Perform research	Research = Intermediate product 2.1	X
QA 2.1	Create evaluation specification	Evaluation specification = Intermediate product 2.2	X
QA 1.3		→ Research review = Review 2	X
PE 3	**Evaluation Phase**		
PE 3.1	Evaluation by example	Evaluation example = Intermediate product 3.1	X
PE 3.2	Perform evaluation	Evaluation protocol = Intermediate product 3.2	X
QA 1.4		Evaluation review = Review 3	X
PE 4	**Acceptance Phase**		
PE 4.1	Compose a recommendation / decision aid	Recommendation = Intermediate product 4.1	X
QA 1.5		→ Acceptance review	X

Table 10-2: Overview of activities and products of the open model with the example of a market overview

11 Assigning Products to Process Model Classes

Table 11-1 shows an overview of the five process model classes and their required products. The amount of each document depends on the project type: Bachelor Thesis, Research Thesis, Study Thesis, or Master Thesis. Before the end of the project, the thesis paper, and the poster must be created.

Project Phases	Document	Software development	System development	Conceptional projects	Open model
		BT/RT/ST/MT	BT/RT/ST/MT	BT/RT/ST/MT	BT/RT/ST/MT
Definition Phase	Project plan	**X**	**X**	**X**	**X**
	System requirements specification	**X**	**X**	**X**	**X**
	Software system model	**X**	-	-	-
	System model	-	**X**	-	-
	Intermediate product 1.1	-	-	-	**O**
	Basis	-	-	**X**	-
Design Phase Conception Phase Project Phase 2	Software system architecture	**X**	-	-	-
	System architecture	-	**X**	-	-
	Intermediate product 2.1	-	-	-	**X**
	Software components specification	**X**	-	-	-
	System components specification	-	**X**	-	-
	Concept	-	-	**X**	-
	Intermediate Product 2.2	-	-	-	**O**
	Test specification	**X**	**X**	-	-
	Evaluation specification	-	-	**O**	-
Impl. or Real. Phase	Prototype description	-	-	**O**	-
	Intermediate product 3.1	-	-	-	**X**
Eval. Phase Project Phase 3	Test protocol	**X**	**X**	-	-
	Evaluation protocol	-	-	**O**	-
	Intermediate product 3.2	-	-	-	**O**
Integration and Acceptance Phase	User manual	**X**	**X**	-	-
	Intermediate product 4.1	-	-	-	**X**
	Intermediate product 4.2	-	-	-	**O**
Project Phase 4	Final project report	**X**	**X**	**X**	**X**
	Abbreviations	**X**	**X**	**X**	**X**
	Terminology	**X**	**X**	**X**	**X**
	Literature	**X**	**X**	**X**	**X**
Minimum Number of Documents		**12**	**12**	**8**	**9**
Maximum Number of Documents		**12**	**12**	**11**	**13**

Table 11-1: Assigning the products to the process model classes

Legend:

BT	Bachelor Theses
RT	Research Theses
ST	Study Theses
MT	Master Theses
X	Document is required
O	Document is optional
-	Document is not required

12 IAS Development Environment (DE)

12.1 Introduction

This document describes all tools that are required for working on Theses of any subtype for the IAS. In addition, the valid programming regulations and directory structures for the IAS are listed.

This document is divided into four chapters. In chapter 12, the IAS development environment and all tools accepted by the IAS for working on software projects[1] in this environment are listed. In chapter 13, the programming regulations are listed, that must be adhered to during the work at the IAS. Chapter 14 describes the IAS directory structure i.e., the position and structure of directories for document storage.

12.2 Software Tools for Standard Development Process at the IAS

Several software tools are available to support software projects at the IAS. These tools are classified in two categories:

1. Standard tools in the development environment
2. Standard tools in the office area

Several software tools are required during the development of software (standard tools in the development environment). Modeling a system during requirements engineering and the design phase is supported by a graphical tool. Likewise, implementation and testing are supported by an implementation environment. The standard tools are selected suitable to the IAS standard development process. In the IAS standard development process the Unified Modeling Language (UML) is used for modeling (standard modeling notation). Additional conventions and details can be found in the section *12.2.1 Standard Tools in the Development Area*. The tools to be used for project documentation are described in section *12.2.2 Standard Tools in the Office Area*.

All software work at IAS takes place on the operating system platform MS Windows 8. In justified exceptions, it is also possible to work with Linux. However, there are no standard tools available for Linux. All documents of the office area should be created under MS Windows 8.

12.2.1 Standard Tools in the Development Area

The standard development tools are divided in two kinds of tools: the standard tools of the modeling environment and the standard tools of the implementation environment. The graphics program VISIO is used for UML modeling. The C/C++ and C# implementation is normally done with the tool Microsoft Visual Studio .net2012. For implementing Java, normally Eclipse is used. Only in reasonable cases, other software tools may be used than the following.

12.2.1.1 Modeling in UML with Visio

When using Visio for UML modeling, the respective UML template of Visio should be used for the different diagrams of UML.

[1] Mostly smaller „one-person“ student projects, especially UPs and MTs.

12.2.1.2 Integrated Development Environments for Implementation (IDE)

Microsoft Visual Studio provides a uniform development surface for C, C++ and C# programming. The Microsoft Visual Studio includes an editor, a compiler, a linker and an extensive online help system. The application can be operated intuitively. All necessary manuals can be found online.
For development of software in Java, the open source platform Eclipse is used at the institute. It contains editors, a compiler and a debugger for the comprehensive support of Java implementation. For the development environment, extensive online documentation and tutorials are available.

12.2.2 Standard Tools in the Office Area

Intermediate results during the development must be documented in text or graphics. If no tools are available in the development area for this purpose, then the standard tools for the office must be used. Products defined in the process model are expected in the form of documents and must also be created with the help of office tools.

The use of the following software products for these tasks is mandatory:

- Microsoft Office
- Microsoft Visio
- Microsoft Project

These software products can be used intuitively and are sufficiently described in the online user documentation. There are online tutorials for the individual software packages that explain the principal operation of these tools. During the learning phase it is recommended that you use these tutorials. No further user documentation is available.

12.3 Available Software at IAS and IAS Standard Installation

Table 2 gives an entire overview on all available software at the IAS. In addition to the standard tools described in section 12.2, there are various special tools available at the IAS. Their deployment depends on the task posed by each project. Because the goal of this documentation is general information only, these tools will not be discussed in detail. Often manuals or references for these tools are available online or in the IAS library. Table 2 lists an overview of the application areas for all tools.

In addition, Table 2 describes which software is installed as standard on every working place computer. These form the IAS standard installation.

Tool type	Tool name	Application and comments	Std.-Inst.[2]
Operating systems and server	MS Windows 8	Operating System	X
	Linux	Operating System	
	Apache	Webserver	
	Virtual Box	Virtualisation of computers	X
Office, communication software and auxiliary tools	MS Office (multilanguage)	Word processing, spreadsheet tool, presentation tool	X
	MS Outlook	E-Mail, Dates, Tasks	X
	MS Visio	Diagrams and graphics	X
	MS Project	Project planning	X
	Gimp	Simple graphic tool	
	Notepad++	Editor with Syntax-Highlighting	X
	Mozilla Firefox	WWW browser	X
	Internet Explorer	WWW browser	X
	7Zip	Data compressor	X
	Acrobat Reader	PDF file viewer	X
Information presentation	IrfanView	Graphic viewer	X
	Camtasia	Tool for creating screen movies	
	VLC Player	Video player	
Configuration management	TortioseGIT	Windows-Client for GIT	X
	TortoiseSVN	Windows-Client for SVN	X
	Beyond Compare	Tool for comparing two files	
Modeling with block charts and state charts	Matlab/Simulink/ Stateflow	Tool for control design, simulation, C/C++ code generation, test	
	LabView	Graphical simulation and development environment	
Ada programming Java programming C/C++ and Visual Basic programming	ObjektAda	Ada development environment	
	Gnat (GNU Ada Compiler)	Ada development environment for different platforms: WinNT, DOS, Linux and others	
	Eclipse	Open Source development environment with Java SDK and Android SDK	X
	Microsoft Visual Studio	C/C++ /C# and Visual Basic development environment for Win32 platforms, Microsoft Foundation Classes (MFC), Active Template Library (ATL) and other Microsoft specific extensions	X
C/C++ programming for micro controllers	Tasking C/C++ Cross Compiler for C16x/ST10	C/C++ compiler for Infineon C16x microcontroller	
	Tasking CrossView Pro Simulator und ROM Monitor Debugger	Simulator and debugger for Infineon C16x microcontroller	
	ATMEL Studio	IDE and Compiler for ATMEL Microcontroller	
	CANoe (Vector)	Development, simulation, visualization and analysis environment for CAN bus systems	
	MPLAB X	IDE and Compiler for Microchip Microcontroller	
SPS programming	MySQL	Database server for Linux and Win32 platforms	

Database development	Sybase SQL Anywhere	Relational database server for Win32 platforms	
	Sybase Powerbuilder	GUI builder, especially designed for database applications	

Table 2: Available software at the IAS

The IAS development environment has thereby any software which is needed for the work in a students' thesis of any subtype. Therefore, no compelling necessity is given that students install their own software on the computers. In order to avoid inconsistencies, students are not allowed to install their own software. If such software is noticed, it is removed. The student does not have guarantee that a, in such a way installed software remains. If additional tools should be needed for the execution of the work, this has to be coordinated with the refering supervisor (see also chapter 12.4 special software on CIP computers).

12.4 Special Software on CIP Computers

Special software is software, which is not defined as standard software in Table 2. This special software is usually needed by one or only few students for the duration of the collegiate work. However, the functionality of the software of the IAS standard installation must not be impaired by the special software. Therefore, it can be ensured that the treatment of the documents of the IAS standard development process can be done on each PC of the PC laboratory.

The procedure for the installation of the special software depends on the respective license model. Is there a floating license available the special software, depending on how many students like to use it at the same time, will be installed on several to all PC of the PC laboratory. Is there only a single place license available, the special software must be installed only on one PC (depending on the number of available licenses). The installation is done by the supervisor, whereby it must be ensured that on each PC only one special software with single place license is installed. An appropriate list at each PC of the PC laboratory documents the up-to-date installed special software.

To minimize side effects through installed special software, PCs with special software have to be installed again with the IAS standard installation, when this software is not used any more. The time of the new installation is specified by the release of the supervisor, who installed the special software. Data in the Exchange folder may not be affected by the new installation. At the latest 4 weeks after the work the PC should be installed again. Exceptions are possible only if the installation is still needed longer for a backup copy etc.

13 Programming Guidelines[3]

13.1 Guidelines for Comments

You can find templates and examples how to comment your source code (e.g. for the programming language Java with Javadoc) in the Online-QM-compenidum under "Studierende|SADA|Unterstützende Vorlagen und Dokumente".

Write your comments only in Englisch. You must comment the source code, so that others can understand its structure and functions. Avoid trivial comments, such as:

```
// wrong
++x; // performs the calculation x = x + 1
```

Add a short description to all global identifiers, structures, macros etc.

13.1.1 Comments for Files (Module, Class, ...)

Add the following information at the beginning of each file:

- file name
- version number in the format <version>, yyyy-mm-dd
  e.g. 1,1, 2006-12-24
- brief description
- detailed description (optional)
- copyright
  e.g. (C) 2003-2014 IAS, Universitaet Stuttgart

13.1.2 Comments for Functions and Methods

Add the following information to each function or method:

- brief description
- detailed description (optional)
- list of parameters with an explanation of each parameter
  if the programming language allows different type of parameters, add a description of this type such as input (in), output (out) or input and output (inout)
- description of the return value if present
- exceptions, which can be thrown by the function (if applicable in the programming language you use)
- further details (optional)

13.2 Guidelines for Identifiers

Choose the name of your identifier, so that it is as brief and meaningful as possible. Use English identifiers only.

The type should be clear from the context. Hence, avoid type prefixes (e.g. cChar) except for pointers, which are preceded by a small p (e.g. pPointer). Self-defined types such as basic classes, structs or type definitions must be preceded by a capital T. For instance:

[3] The guidelines are kept generic and must be applied to all programming languages (e.g. Java, C#, C++, C, ...)

```
class TName
{
      String firstName;
      String lastName;
}
```

Use capital and small letters as follows:

- Two identifiers must not only differ in small/capital letters (e.g. MyClass and mYClaSs)
- If an identifier consists of several words, begin each word with a capital letter. If you use abbreviations or acronyms write only the first letter as capital letter, e.g. `HttpInitialize`.
- Write names of Macros exclusively in capital letters and digits. Separate words by using an underscore, e.g. MY_EXAMPLE_MACRO_1. Constants are named respectively.

In programming languages, which are not object-oriented (e.g. C), begin local identifiers with an underscore. This applies to names of variables and functions. In all other cases underscores are not allowed at the beginning of an identifier.

The following table gives an overview of possible identifiers:

Classes, Interfaces, Structs	Use nouns as identifier, which start with a capital letter (e.g. ThisIsAClass). Self-defined types begin with a capital „T“.
Methods, Functions	Use verbs. Names of functions or methods begin with a small letter (e.g. getReturnValue). Methods for reading or writing instance variables should begin with set or get.
Variables	Write variable names in small letters. This also applies to structs or class attributes. Name counter variables i, j, k, and so on. Boolean variables should be named using participles or isXy (e.g. found, isValid).

13.3 Formatting Guidelines

The number of characters per line should not exceed 80.

Use blank lines to increase the readability of the source code. Use as many blank lines as necessary and as few as possible. We recommend separating blocks, that logically belong together, by blank lines. Add blank lines in front of function declarations.

Indent code blocks ("{" - "}", BEGIN - END). Use spaces instead of tabs. Often editors can be configured, so that tabs are automatically replaced. Use the settings

- Set tab width to „4“

- Activate „Replace tabs“

Example:

```
if (condition1)
{
      statement1;
      if (condition2)
          statement2;
      else
          statement3;
}
else
{
      statement4;
      statement5;
}
do
{
      statement;
} while (condition);
```

The formatting guidelines assure that the source code is formatted in a standardized way independent of the programmer.

13.4 Programming Style

The number of parameters and local variables of a function/method should be only as high as necessary. We recommend not declaring more than 5 parameters and not more than 7 local variables (including parameters).

A function/method should exceed a length of 70-100 lines of code, only if necessary (e.g. a further division makes no sense).

Functions and variables, which are required within a single source file, should be declared as ‘static’ (applies to the programming language C).

Avoid side effects, which may e.g. occur in if-clauses. Example:

```
// wrong
while (i++ < 50 && (j = get()) != 0)
{
     …
}

// right
j = get();
while ((i < 50) && (j != 0))
{
      ++i;
      j = get();
}
```

Initialize local, non-static variables before the first use and NOT together with the declaration.

Do NOT use GOTO.

Avoid constant number in the source code. Instead use constants, enumerations or defines with meaningful name, e.g. `#define MAX_COUNT 5`.

13.5 Links

Guidelines for commenting Java code using Javadoc
http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

We recommend using the Javadoc style guides
http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#styleguide

14 Directory Structure

The directories are physically located on a centralized file server. Each student can access his home directory via the Z:\ drive. (Note: Data on the centralized file server is saved daily and can be restored if needed.)

There are three main directories for data that are described in the following chapters:

- Student's own directory (full access)
  \\ias-cifs.tiks.uni-stuttgart.de\de\ias\stud-home\<login>
- Project directory for Theses (read-only access for students)
  \\ias-cifs.tik.uni-stuttgart.de\ias\sadapro
- Template directory for Theses (read-only access for students)
  \\ias-cifs.tik.uni-stuttgart.de\ias\studdir\Templates

14.1 Student Directory on Fileserver

Within the student's own directory, the following structure can be found:

.../**<login>**

work (work directory with the same structure as sadapro-dir)

The name for an upmt[4] project directory is assembled as follows:

<sadapro-dir> = [„mt“ | „rt“ | „st“ | „bt“] + „-“ + <pro-nr> + <lastname>
<pro-nr> = /* Project number as determined upon applying at IAS e.g., 1536 */
<lastname> = /* Last name in small letters*/
mt: master thesis; rt: research thesis; st: study thesis; bt: bachelor thesis

14.2 Sadapro Project Directory

The project directory serves as storage space for all **released** documents. All versions of **released** documents are stored in the respective subdirectories of the directory **<sadapro-dir>/<upmt project directory name>** as follows. The syntax for file name assignment is: <filename>-v<XX>.<extension>. The version counter <XX> begins with 10. The following versions are numbered in succession and stored in the same directory. The file with the highest version number is the most current. It is easy to ascertain the current version manually or with software. Example:

[4] Undergraduate Project and Master Thesis

sw-system-requirements-specification-v10.docx
sw-system-requirements-specification-v11.docx
sw-system-requirements-specification-v12.docx
sw-system-requirements-specification-v13.docx (most current Version)

The version-based storage of all documents lies within the responsibility of the student and must be performed manually.

Documents are stored for each process model class in the same directory structure. For the different process model class, the names of the documents differ.
Project directory structure of process model class „software development“

- Project directory structure of process model class „system development“
- Project directory structure of process model class „conceptional project“
- Project directory structure of process model class „open model”

14.2.1 Project Directory Structure for all Process Model Classes

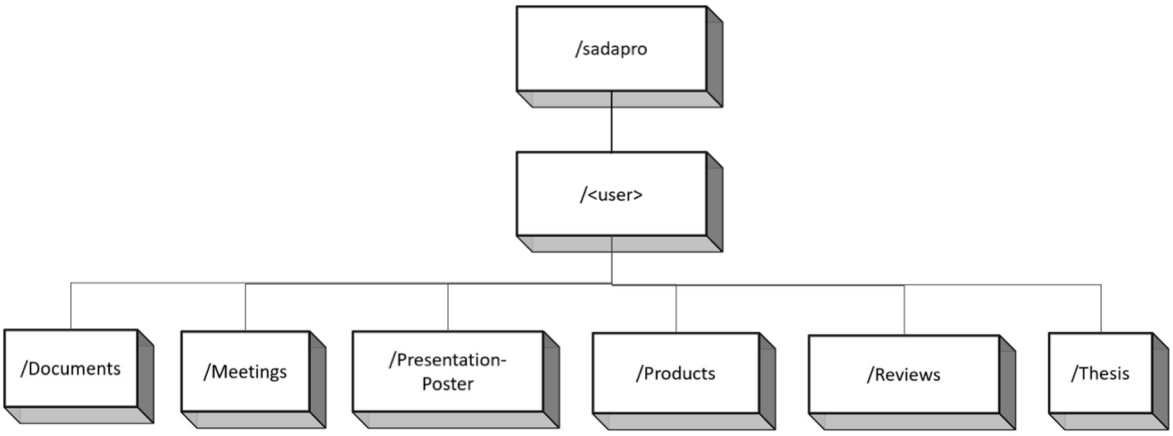

Figure 2: Project directory structure sadapro for all PM classes

14.2.2 Directory Structure for Process Model Class „Software Development“

The general structure including the respective file names of the sadapro project directory are as follows:

\\ias-cifs.tik.uni-stuttgart.de\ias\sadapro

Documents
Abbreviations.docx
Final Report.docx
Glossary.docs
Literature.docx
sw-Component Specification.docx
sw-Project Plan.docx
sw-System Architecture.docx
sw-System Modell.docx
sw-System Requirements Specification.docx
sw-Test Protocol.docx
sw-Test Specification.docx

sw-User Manual.docx
sw-User Requirements Specification.docx

Meetings
Kick-off Meeting.pptx
Intermediate Report.pptx

Presentation-Poster
Poster.pptx
Thesis Presentation.pptx

Products
<sourcecode>.zip

Reviews
sw-Review.docx

Thesis
Thesis.docx

14.2.3 Directory Structure for Process Model Class „System Development“

The general structure including the respective file names of the sadapro project directory are as follows:

\\ias-cifs.tik.uni-stuttgart.de\ias\sadapro

Documents
Abbreviations.docx
Final Report.docx
Glossary.docs
Literature.docx
sy-Component Specification.docx
sy-Project Plan.docx
sy-System Architecture.docx
sy-System Modell.docx
sy-System Requirements Specification.docx
sy-Test Protocol.docx
sy-Test Specification.docx
sy-User Manual.docx
sy-User Requirements Specification.docx

Meetings
Kick-off Meeting.pptx
Intermediate Report.pptx

Presentation-Poster
Poster.pptx
Thesis Presentation.pptx

Products
<sourcecode>.zip

Reviews
sy-Review.docx

Thesis
Thesis.docx

14.2.4 Directory Structure for Process Model Class „Conceptional Projects“

The general structure including the respective file names of the sadapro project directory are as follows:

\\ias-cifs.tik.uni-stuttgart.de\ias\sadapro

Documents
Abbreviations.docx
Final Report.docx
Glossary.docs
Literature.docx
co-Basis.docx
co-Conception.docx
co-Evaluation Protocol.docx
co-Evaluation Specification.docx
co-Project Plan.docx
co-Prototype Description.docx
co-System Requirements Specification.docx
co-User Requirements Specification.docx

Meetings
Kick-off Meeting.pptx
Intermediate Report.pptx

Presentation-Poster
Poster.pptx
Thesis Presentation.pptx

Products
<sourcecode>.zip

Reviews
co-Review.docx

Thesis
Thesis.docx

14.2.5 Directory Structure for Process Model Class „Open Model“

The general structure including the respective file names of the sadapro project directory are as follows:

\\ias-cifs.tik.uni-stuttgart.de\ias\sadapro

Documents
- Abbreviations.docx
- Final Report.docx
- Glossary.docs
- Literature.docx
- om-Intermediate Product 1-1.docx
- om-Intermediate Product 2-1.docx
- om-Intermediate Product 2-2.docx
- om-Intermediate Product 3-1.docx
- om-Intermediate Product 3-2.docx
- om-Intermediate Product 4-1.docx
- om-Intermediate Product 4-2.docx
- om-Project Plan.docx
- om-System Requirements Specification.docx
- om-User Requirements Specification.docx

Meetings
- Kick-off Meeting.pptx
- Intermediate Report.pptx

Presentation-Poster
- Poster.pptx
- Thesis Presentation.pptx

Products
- <sourcecode>.zip

Reviews
- om-Review.docx

Thesis
- Thesis.docx

14.3 Template Directory

The template directory contains the templates for documents that are created during the work on the project. The type of template to be used can be interpreted from the file name.

14.4 Initializing the Directory Structure

The student’s directory and the UPMT project directory together with their complete structures are automatically created when a new user (student) is inserted by the system manager (without documents).

15 Saving the Data Upon Finishing the Thesis

It is strongly recommended to save the whole directory to a private USB stick upon finishing the thesis. Thus, the access to the data can still be guaranteed after the IAS account gets deleted.